KERNEL METHODS IN STEGANALYSIS

BY

TOMÁŠ PEVNÝ

M.S. in Computer Science, Czech Technical University, Prague, 2003

DISSERTATION

Submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in Computer Science
in the Graduate School of Binghamton University
State University of New York
2008

Accepted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in Computer Science
in the Graduate School of Binghamton University
State University of New York
2008

April 30, 2008

Jessica Fridrich, Department of Electrical and Computer Engineering,
Binghamton University

Mark Fowler, Department of Electrical and Computer Engineering,
Binghamton University

Walker Land, Department of Bioengineering,
Binghamton University

Mark Zhongfei Zhang, Department of Computer Science,
Binghamton University

Abstract

Steganography is the art of covert communication. Its goal is achieved by hiding secret messages into innocuous objects such as digital images, audio files, etc. As any other technology, steganography can be used for malicious purpose. The need to detect steganographic objects give rise to steganalysis, a complementary task to steganography. The main focus of this dissertation is on steganalysis of JPEG images, as the JPEG image format is nowadays the most frequently used image format.

In the first part of the dissertation, a detector recognizing six popular steganographic algorithms is presented. This detector is novel in many ways. First, it not only detects the presence of a secret message, but also assigns the image to a class according to the algorithm used to embed the message. Second, it detects stego content in single-compressed as well as in double-compressed JPEG images. Third, it offers superior accuracy in comparison to prior art.

The detector is based on feature extraction and supervised training of two banks of multi-classifiers implemented by Support Vector Machines. The first bank targeted to single-compressed images contains a separate multi-classifier trained for each JPEG quality factor from a certain range. Another bank of multi-classifiers is trained for double-compressed images for the same range of primary quality factors. Multi-classifier banks are preceded by a pre-classifier detecting double-compression and estimating the primary quantization table.

The second part of the dissertation presents a method for benchmarking security of steganographic algorithms. We argue that a good benchmark should be dependent only on the model chosen to represent cover and stego objects (feature set). While the KL divergence would be a preferable measure, because it is a fundamental quantity, there are practical difficulties in computing it. Therefore a Maximum Mean Discrepancy (MMD) is proposed as a measure of steganographic security, because it is well understood theoretically, and is numerically stable even in high-dimensional spaces.

Acknowledgments

There are many people to thank for the support and help. But I would like to particularly thank my advisor, Professor Jessica Fridrich , for countless hours spent guiding me, discussing ideas, and reviewing my work.

I would like to thank the rest of the committee, Professor Mark Fowler, Professor Walker Land, and Professor Zongfei Zhang for their advice and patience.

Also, I would like to express my thanks to my parents, my family, teammates, and friends for their great support and help.

Special thanks belong to the Air Force Research Laboratory, particularly to Richard Simard, for supporting of our research.

Without all those, this work will never be done.

Contents

Abstract vii
Acknowledgments ix

List of Tables xiii

List of Figures xv

Chapter 1. Introduction 1
1.1. Steganography 1
1.2. Steganalysis 2
1.3. Practical Steganalysis 4
1.4. Dissertation Goals and Outline 7

Part 1. Blind steganalyzer for JPEG images 9

Chapter 2. Outline of the Blind Steganalyzer 11
2.1. Goals and design choices 11
2.2. Prior Art 12
2.3. Outline of the proposed Blind Steganalyzer 13
2.4. Basics of JPEG Compression 15
2.5. Database of Images 17

Chapter 3. Detection of double-compression 19
3.1. Motivation 19
3.2. The Proposed Approach 23
3.3. Experimental results 25

Chapter 4. Feature set for Blind Steganalysis 33
4.1. Calibration 33
4.2. Original DCT feature set and Extended DCT feature set 34
4.3. Original and Calibrated Markov features 37
4.4. Merged feature set 38
4.5. Comparison of features 39

Chapter 5. Blind Steganalyzer for JPEG Images 43
5.1. Classifier for Single-Compressed JPEG images 43
5.2. Classifier for Double-Compressed JPEG images 45
5.3. Experimental results from the Blind Steganalyzer 45
5.4. Conclusion 53

Chapter 6. Novelty detection in Steganalysis 55
6.1. Novelty detection: an overview 56
6.2. Experimental comparison 59
6.3. Conclusions 63

Chapter 7. Alternative use of blind steganalysis feature sets 65
7.1. Targeted steganalysis 65
7.2. Dimensionality reduction 66
7.3. Practical benchmarking of steganographic schemes 67
7.4. Steganography design 67

Part 2. Security of steganographic schemes 69

Chapter 8. Revisited Security of Steganographic Scheme 71
8.1. KL-Divergence 74
8.2. Maximum Mean Discrepancy (MMD) 77
8.3. Experiments 84
8.4. Conclusions 87

Appendix A. Support Vector Machines 91
A.1. Linear Support Vector Machines 91
A.2. Kernelized Support Vector Machines 97
A.3. Weighted Support Vector Machines 98
A.4. Practical use of Support Vector Machines 99
A.5. Multi-classification 103

Bibliography 107

List of Tables

3.1 Primary quantization steps (PQS) detectable by the multi-classifier for a given secondary quantization step (SQS). The last column (#SVMs) shows the number of binary Support Vector Machines in the multi-classifier. 25

3.2 Accuracy of double-compression detectors employing Benford and Multiple-counting features. Since Benford features are not designed to deal with stego images, both detectors were trained and tested on *cover* images only. 28

3.3 Accuracy of Neural Network (NN) and Support Vector Machine (SVM) primary quantization steps detectors on *COVER* images from the testing set. PQS and SQS stand for primary and secondary quantization steps, respectively. 29

3.4 Accuracy of Neural Network (NN) and Support Vector Machine (SVM) primary quantization steps detectors on *COVER* and *STEGO* images from the testing set. PQS and SQS stand for primary and secondary quantization steps, respectively. 29

3.5 Mean and standard deviation of the drop in accuracy of the Primary Quality Factor estimator when applying it only to stego images and only to cover images. 32

4.1 Extended DCT feature set with 193 features. 37

5.1 Confusion matrix for the multi-classifier trained for quality factor 75 tested on single and double-compressed 75-quality JPEG images. The first column contains the embedding algorithm and the relative message length. The remaining columns show the classification results. JP H&S is an abbreviation of JP Hide&Seek. 50

5.2 Confusion matrix for the multi-classifier trained for quality factor 80 tested on single and double-compressed 80-quality JPEG images. The first column contains the embedding algorithm and the relative message length. The remaining columns show the classification results. JP H&S is an abbreviation of JP Hide&Seek. 51

5.3 Confusion matrix for stego images with non-standard quantization tables from Fuji E550. JP H&S is an abbreviation of JP Hide&Seek. 51

5.4 Confusion matrix of single-compression multi-classifier $\mathcal{S}_{75}$ (Section 5.1) calculated on double-compressed images from the testing set. If we compare accuracy in this Table with accuracy of the blind steganalyzer correctly handling double-compressed images (Table 5.1), we can see that increased accuracy of the blind steganalyzer justifies its increased complexity. JP H&S is an abbreviation of JP Hide&Seek. 52

5.5 Confusion table of the multi-classifier on images from the testing set embedded by Jsteg, MMx, and –F5. The multi-class detector was not trained to detect any of these algorithms. JP H&S is an abbreviation of JP Hide&Seek. 53

6.1 Comparison of accuracy of general steganography detectors on “unknown” algorithms. The detector “OC-SVM$_{\text{shift}}$” is an OC-SVM classifier with the threshold shifted to match the false positive rate of the OC-NM classifier. 60

6.2 Comparison of accuracy of general steganography detectors on "known" algorithms. Note that except for the detector DLD-SVM$_{\text{loc}}$, these algorithms were not used to create images in the training set, which makes them "unknown." The detector "OC-SVM$_{\text{shift}}$" is an OC-SVM classifier with the threshold shifted to match the false positive rate of the OC-NM classifier. 61

6.3 Percentage of processed covers detected correctly as covers. 62

6.4 Percentage of correctly classified covers that were double-compressed using primary quality factor PQF and secondary quality factor 75. 62

7.1 Detection accuracy of targeted steganalyzers calculated on images from testing set. All steganalyzers are soft-margin C-SVMs. The message length of stego images embedded by MMx correspond to the maximal messages for Hamming codes (1,3,2), (1,7,3), and (1,15,4)). 65

8.1 Relative error of the KL-divergence estimate for two multi-variate Gaussian distributions for various combinations of sample sizes, l, and data dimensionality d. The number of nearest neighbors was set to $k = 1$ in order to achieve minimal bias of the estimator. 76

8.2 Relative error of kNN estimators of KL-divergence estimate on multi-variate Gaussian distributions with data dimensionality $d \in \{1, 10, 100\}$ and converging means. The last column denoted $\mu = 0$ shows an absolute value of estimates for the case, when both distributions are the same ($D_{\text{KL}}(p_c, p_s) = 0$). Estimates were calculated from 2×10^5 samples. The number of nearest neighbors was set to $k = 1$ in order to achieve minimal bias of the estimator. 76

8.3 Relative error of sample $\text{MMD}_u^2(\mathcal{F}, \mathbf{X}, \mathbf{Y})$ with Gaussian kernel between two d-dimensional multivariate Gaussian distributions $N(-\frac{1}{\sqrt{d}}, \mathbf{I})$ and $N(\frac{1}{\sqrt{d}}, \mathbf{I})$ calculated from l data samples in d-dimensional space. The rightmost column denoted as ∞ shows the true value of MMD. The second leftmost column shows the width of Gaussian kernel γ used to calculate the estimates and true value. 85

8.4 Relative error of MMD estimators on multivariate Gaussian distributions with dimensionality $d \in \{1, 10, 100\}$ and converging means. Estimates were calculated from 2×5000 samples. The second leftmost column shows the width of Gaussian kernel used to calculate estimated in the same row. 86

8.5 MMD calculated between two multi-dimensional Student distributions $p_c \sim p_{\text{St}}(\lambda = 0.01, \nu = 2, \mu = 0)$ and $p_s \sim p_S(\lambda = 0.01, \nu = 2, \mu = \frac{0.1}{\sqrt{d}})$, where $d \in \{1, 5, 10, 100, 200, 300\}$ is the dimension of the problem. 86

List of Figures

1.1 Prisoner's problem: Alice wants to send Bob a secret message containing an escape plan from the prison. The message is intercepted by warden Eve. If Eve notices any traces of secret data, she will put Alice and Bob into solitary confinement without any means of communication. Their chances to escape from the prison would be ruined. 3

2.1 Feature space spanned by eigenvectors of the three largest eigenvalues after principle component transformation of Merged features (Chapter 4). Features are extracted from cover images and images fully embedded by F5, JP Hide&Seek, MBS1, MBS2, OutGuess and Steghide. 13

2.1 Outline of the blind steganalyzer for JPEG images. 14

3.1 Effect of double-compression on histograms of absolute values of DCT coefficients on a fixed mode $(0, 1)$. The secondary quantization step is in all four cases the same, $Q^2_{ij} = 4$, only the primary quantization step Q^1_{ij} varies. Figure (a) shows histogram of single-compressed DCT coefficients, Figure (b) shows histogram exhibiting zeros, Figures (c) and (d) show histogram exhibiting double-peaks, and, finally, Figure (e) shows histogram exhibiting divisor effect — histogram is not influenced by double-compression. 21

3.1 Accuracy of the double-compression detector for secondary quality factors 75 and 80 on double-compressed cover images and images embedded with F5 and OutGuess algorithms. Images with primary quality factor equal to the secondary quality factor are not double-compressed, which means that in this case, the correct answer of the detector is single-compressed. Graphs are drawn with respect to the primary quality factor. 27

3.2 Accuracy of the double-compression detector on single-compressed JPEG images with quality factors 75 and 80. Note that the range of Y axis is $[90\%, 100\%]$. 28

3.3 Accuracy of primary quality factor estimator for secondary quality factors 75 and 80 on double-compressed cover images and images embedded with F5 and OutGuess algorithms. Graphs are drawn with respect to the true primary quality factor. The graph showing OutGuess images with secondary quality factor 80 starts from the primary quality factor 70 because OutGuess fails to embed message into images with combination of primary quality factors $63, \ldots, 69$ and secondary quality factor 80. 31

3.4 Quantization steps of standard quantization matrices from quality factors $75, 88$, and 89 on modes from the set $\mathcal{L}$ 32

4.1 Calibration of the single-compressed stego image. Stego JPEG image J is decompressed to the spatial domain, cropped by a few pixels in both directions, and compressed again with the same quantization matrix as the original stego image J. 33

4.2 Calibration of double-compressed image. The stego image J is decompressed to the spatial domain, cropped and compressed with the primary (cover) quantization matrix $\hat{Q}^{(1)}$. This new image $J^{'}$ is the estimate of the image before embedding. $J^{'}$ is again decompressed and compressed with the secondary quantization matrix $Q^{(2)}$, which yields into calibrated image $\hat{J}$. 34

4.1 Schematic of the formation of difference array $F_h(u, v)$ along horizontal axis. S_u denotes width of the JPEG image in pixels. Arrows depicted in the difference array matrix $F_h(u, v)$ show the direction of the transition probabilities in the matrix $\mathbf{M}_h$. 37

4.1 Comparison of DCT, extended DCT, original Markov, calibrated Markov, and Merged feature sets on the problem of detection of particular steganographic algorithm (binary classification). The error is measured as $P_{err} = 0.5(P_{FP} + P_{MD})$ on images from the testing set. All images are single-compressed JPEG images with quality factor 75. 39

4.2 Comparison of DCT, extended DCT, original Markov, calibrated Markov, and Merged feature sets on multi-classification problem of detection of steganographic algorithm. Graphs shows detection accuracy, which is the rate of correctly assigning an image to the class according to the algorithm used for embedding, calculated on images from the testing set. All images are single-compressed JPEG images with quality factor 75. 41

5.1 Detection accuracy of the blind steganalyzer on single-compressed JPEG images. Detection accuracy is plot with respect to quality factor of JPEG images. Note the different y-axis scale for JP Hide&Seek. 46

5.2 Detection accuracy of the multi-classifier on single-compressed cover images from the testing set. 47

5.3 Accuracy of the multi-classifier on double-compressed JPEG images with secondary quality factors 75 and 80. The graph showing OutGuess images with secondary quality factor 80 starts from the primary quality factor 70 because OutGuess fails to embed message into images with combination of primary quality factors $63, \ldots, 69$ and secondary quality factor 80. 48

5.4 Accuracy of the multi-classifier on double-compressed cover JPEG images with secondary quality factors 75 and 80. 49

8.1 MMD (left) and probability of error for an SVM (right) for 10 steganographic algorithms and 5 payloads. To obtain a better visual correspondence between the graphs, we show $-\log_{10} \text{MMD}[\mathcal{F}, \mathbf{X}, \mathbf{Y}]$. The horizontal lines indicate the threshold of undetectability determined as MMD from two samples of covers. Algorithms with MMD close to the line are recognized as secure with respect to the given set of features. 88

A.1 Example of linearly separable training set in $\mathcal{X} = \mathbb{R}^2$. Separating hyperplane (thick solid black line) is defined by the examples identified by dashed circle (called support vectors). Notice that other examples do not affect the solution of the optimization problem. By using notation borrowed from the text, for points on the separating hyperplane holds $\mathbf{w}^* \cdot \mathbf{x} - b^* = 0$, for points on dashed lines (support vectors) holds $\frac{\mathbf{w}^*}{\epsilon} \cdot \mathbf{x}^\circ - \frac{b^*}{\epsilon} = +1$ and $\frac{\mathbf{w}^*}{\epsilon} \cdot \mathbf{x}^\circ - \frac{b^*}{\epsilon} = +1$. For distance between support vectors and separating hyperplane holds $\left|\frac{\mathbf{w}^*}{\|\mathbf{w}^*\|} \cdot \mathbf{x} - \frac{b^*}{\|\mathbf{w}^*\|}\right| = \frac{\epsilon}{\|\mathbf{w}^*\|}$. 93

A.2 Example of linearly non-separable training set on $\mathcal{X} = \mathbb{R}^2$. Separating hyperplane (thick solid black line) is defined by the examples identified by support vectors identified by dashed circle. Incorrectly classified examples are identified by cross $\times$ together with slack variables $\xi^{\bullet}, \xi^{\circ} > 0$. 94

A.3 Comparison of step function $u(x) = \begin{cases} 0 & x < 0 \\ 1 & \text{otherwise,} \end{cases}$ and its convex majority hinge loss $h(x) = \max\{0, 1 + x\}$. 95

A.1 Example of ROC curve created by varying threshold b^* of Support Vector Machine. 101

A.1 The decision tree of DAG-SVM for finding the best out of four classes. Equivalent list of states is shown next to each node. 105

CHAPTER 1

Introduction

1.1. Steganography

Steganography is the art of invisible communication between trusted parties. Its purpose is to hide from untrusted parties the fact that any *secret message* is being communicated. This is quite different from *cryptography* trying to make the content of the communication inaccessible, but it is apparent that a secret communication is taking place. Sound difference between cryptography and steganography is that cryptography provides *privacy*, while steganography provides *secrecy*.

The origins of steganography dates back to ancient times. Herodotus in "The Histories of Herodotus" mentions two examples of steganography. Demeratus, a Greek exile in Persia warned Spartans that Xerxes is preparing to invade Greek by writing the message on the backing of a wax tablet and covering it with fresh layer of wax. The ruler of Miletus Histaeus sent message to his friend Aristagorus about urging revolt against Persians, when he shaved head of his most trusted slave, tattooed the message on slave's scalp, and waited till the hair grew back.

Throughout the centuries steganographic methods evolved. Nowadays, the communicated message is embedded into innocuous *cover* media such as digital images, audio and video files, printed documents, etc., by means of subtle changes in the cover media. This dissertation is targeted to steganography in digital images. The next examples show the reported use of steganography by malicious subjects, which emphasize the increasing importance of steganalysis.

- The USA Today on 5th February 2001 published two articles titled "Terrorist instructions hidden on-line" and "Terror groups hide behind Web encryption". Articles said that al-Qaeda operatives have been sending hundreds of encrypted messages that have been hidden in files on digital photographs on the auction site eBay.com.
- The Italian newspaper Corriere della Sera reported that an Al Qaeda cell which had been captured at the Via Quaranta mosque in Milan had pornographic images on their computers, and that these images had been used to hide secret messages (although no other Italian paper ever covered the story).
- The Federal Plan for Cyber Security and Information Assurance Research and Development [8], published in April 2006 makes the following statements:
  - "...immediate concerns also include the use of cyberspace for covert communications, particularly by terrorists but also by foreign intelligence services; espionage against sensitive but poorly defended data in government and industry systems; subversion by insiders, including vendors and contractors; criminal activity, primarily involving fraud and theft of financial or identity information, by hackers and organized crime groups..." (p 9–10)

  - ”International interest in R&D for steganography technologies and their commercialization and application has exploded in recent years. These technologies pose a potential threat to national security. Because steganography secretly embeds additional, and nearly undetectable, information content in digital products, the potential for covert dissemination of malicious software, mobile code, or information is great.” (p 41–42)
  - ”The threat posed by steganography has been documented in numerous intelligence reports.” (p 42)
- Probably the most alarming example of contemporary use of steganography was provided by the captured terrorist training manual, the ”Technical Mujahid, a Training Manual for Jihadis” containing a section entitled ”Covert Communications and Hiding Secrets Inside Images.” It provides details how to practically use steganography for secret communication.

As the steganographic techniques progressed, the requirement from invisibility by human eye evolved into a stronger requirement of statistical undetectability. Before we proceed to the modern definition of steganographic security introduced by Cachin [**9**], we need to describe some notation that will be used throughout this dissertation.

Let $\mathcal{C}$ denote the set of all cover objects c, $\mathcal{M}$ denote the set of secret messages, and $\mathcal{K}$ denote the set of stego keys. A steganographic scheme (algorithm) is a pair $(S_{\mathrm{E}}, S_{\mathrm{X}})$, where $S_{\mathrm{E}} : \mathcal{C}\times\mathcal{M}\times\mathcal{K} \mapsto \mathcal{C}$ is an embedding function and $S_{\mathrm{X}} : \mathcal{C}\times\mathcal{K} \mapsto \mathcal{M}$ is an extraction function. The embedding function S_{E} assigns a new (stego) object $s \in \mathcal{C}$ to each combination of a cover object $c \in \mathcal{C}$, message $m \in \mathcal{M}$, and stego key $k \in \mathcal{K}$. Similarly, the extracting function S_{X} assigns the embedded message m to a combination of stego object s and secret key k.

Let us assume that the probability distribution with distribution function P_c of selecting cover object $c \in \mathcal{C}$ exists. If the secret key $k \in \mathcal{K}$ and message $m \in \mathcal{M}$ are chosen randomly (uniformly), then steganographic the scheme $(S_{\mathrm{E}}, S_{\mathrm{X}})$ together with the pdf of cover images P_c defines probability distribution of stego objects $\mathsf{s} \in \mathcal{C}$ with pdf P_s.

Definition 1.1.1. Steganographic scheme (algorithm) is secure, if the Kullback–Leibler divergence

$$D(P_c||P_s) = \sum_{c\in\mathcal{C}} P_c(c) \log \frac{P_c(c)}{P_s(c)} \tag{1.1.1}$$

between probability distribution of cover objects P_c and probability distribution of stego objects P_s is zero. When $D(P_c||P_s) < \epsilon$, the stego scheme is called ϵ-secure.

A known result of detection theory states that the KL divergence provides an upper bound on the best possible detector one can build [**12**], which implies that secure steganographic algorithm is undetectable. Even though the definition of steganographic security is sound from the theoretical point of view, it is hard to use in practice. The space of all images $\mathcal{C}$ is too large (potentially infinite), which makes the sum in (1.1.1) impossible to calculate with current computers and image models.

1.2. Steganalysis

The counterpart of steganography is *steganalysis* whose goal is discovering the presence of hidden messages. The objective of steganography and steganalysis is modeled by the prisoners' problem (Figure 1.1). Alice and Bob are two prisoners in separated cells and they want to agree on an escape plan. They are allowed to communicate but all messages they exchange are closely monitored by the warden Eve looking for traces of secret data that may be hidden in the objects that Alice and Bob are exchanging. If Eve observers any trace of secret communication, Alice and Bob will be put into solitary confinement

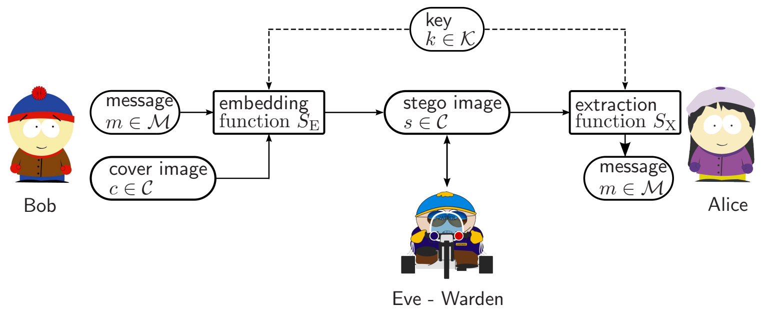


Figure 1.1. Prisoner's problem: Alice wants to send Bob a secret message containing an escape plan from the prison. The message is intercepted by warden Eve. If Eve notices any traces of secret data, she will put Alice and Bob into solitary confinement without any means of communication. Their chances to escape from the prison would be ruined.

without any means of communication. Their chances to escape from the prison would be diminished. Therefore Alice and Bob use steganography to hide details about the escape plan.

Eve's activity is called *steganalysis* and it is a complementary task to steganography. In theory, the steganalyst is successful in attacking the steganographic channel (i.e., the steganography has been broken) if she can distinguish between cover and stego objects with probability better than random guessing. Note that, in contrast to cryptanalysis, it is not necessary to read the secret message to break a steganographic system. The important task of extracting the secret message from an image once it is known to contain secretly embedded data belongs to forensic steganalysis.

Eve can be either *passive-warden* or *active-warden*. Being passive warden means to Eve that she can only observe the communicated objects, while being active warden means that she can also change communicated objects. This work exclusively deals with passive warden scenario.

Under the Kerckhoffs' principle, Eve knows all details (steganographic algorithm, probability distribution on cover objects, etc.) about the communication channel between Alice and Bob except the stego key $k \in \mathcal{K}$. In this case, the security of the steganographic scheme relies entirely on the stego key $k \in \mathcal{K}$. This is the most difficult situation for Alice and Bob. Even though this scenario rarely happens in practice, it should be assumed during the design of a new steganographic algorithm.

In practice, the degree of Eve's knowledge varies. A common scenario is when Eve does not know anything about the cover objects, communicated messages, and the used steganographic algorithm. An example can be an automatic traffic monitoring device (firewall connecting private network to the Internet, or program inspecting images to be posted on a binary discussion group) analyzing all images going through it for hidden messages. This is the most difficult situation for Eve, as she needs steganalysis algorithms capable of detecting as wide spectrum of steganographic schemes as possible.

In some cases, Eve have some side information, which gives her a better chance to detect the presence of steganography. For example Eve can spot Bob downloading steganographic tool from the Internet. This gives Eve knowledge about embedding algorithm that might Alice and Bob be using. Or, if Eve knows that Alice is sending to Bob images from her camera, she can purchase the same camera model as Alice has and tailor the attack to this cover source.

Eve's attack on the stego system can be performed along two lines. She can either inspect statistical properties of cover objects and detect anomalies incompatible with natural (cover) objects, or she can rely on some weakness of the implementation of the steganographic algorithm such as insufficient size of the stego key space or presence of unusual data inserted into the image header by the stego software. While the first kind of attack (called statistical steganalysis) can potentially detect more than one steganographic scheme, the second kind of attacks (called system attacks) are oftenly tailored for a specific implementation of steganographic scheme. This dissertation deals exclusively with statistical steganalysis.

1.3. Practical Steganalysis

This section serves as an introduction to practical steganalysis by emphasizing design decisions Eve needs to make in order to construct a steganalyzer.

1.3.1. Steganalysis from the point of view of Detection Theory. The goal of steganalysis can be defined as a detection problem. Depending on Eve's knowledge about the steganographic scheme used by Alice and Bob, the steganalysis is formulated either as a simple or as a composite hypothesis testing.

If Eve does not know anything about the steganographic scheme, the steganalysis is called *blind* or *universal.* The problem can be formulated as *composite* hypothesis testing

$$\begin{aligned} H_0 &: \mathbf{x} \sim P_c \\ H_1 &: \mathbf{x} \nsim P_c. \end{aligned} \tag{1.3.1}$$

If Eve knows the steganographic algorithm, the steganalysis is called *targeted.* Assuming Eve knows the probability distribution of cover images P_c, steganographic scheme $(S_{\mathrm{E}}, S_{\mathrm{X}})$, and the distribution of the messages, she can theoretically calculate probability distribution of stego images P_s. This additional knowledge in the form of P_s gives her opportunity to tailor her attack specifically to the particular algorithm and get a better detector. The problem of targeted steganalysis is formulated as a *simple* hypothesis testing

$$\begin{aligned} H_0 &: \mathbf{x} \sim P_c \\ H_1 &: \mathbf{x} \sim P_s. \end{aligned} \tag{1.3.2}$$

The formulation of the steganalysis by means of detection theory exposes the fundamental difference between blind and targeted steganalysis. Targeted steganalysis knows probability distribution functions on cover and stego images, which not only gives Eve an opportunity to have a better detector, but she can design the detector to be Neyman-Pearson or Bayesian optimal (the latter only if she knows prior probabilities of encountering cover and stego objects). On the other hand, Blind Steganalysis is a composite hypothesis testing for which optimal detectors may not be known.

1.3.2. Steganographic detectors. Steganographic detector can be described as a mapping $F : \mathcal{C} \mapsto \{0, 1\}$, where $F(\mathbf{x}) = 0$ means that object $\mathbf{x}$ is detected as cover, while $F(\mathbf{x}) = 1$ means that $\mathbf{x}$ is detected as stego. The *critical region* is set $\mathcal{R}_1 = \{\mathbf{x} \in \mathcal{C} | F(\mathbf{x}) = 1\}$, where the detector detects object $\mathbf{x}$ as stego objects. The critical region $\mathcal{R}_1$ fully describes the detector.

There are two types of error the detector F can make. It can either detect a cover object as stego one, which is called false positive (or Type I error), or it can detect a stego object as cover one, which is called missed detection (or false negative, Type II error).

The probability of false alarm P_{FA} or missed detection P_{MD} for a given detector F can be mathematically expressed as

$$\begin{aligned} P_{\mathrm{FA}} &= \Pr\left(F(\mathbf{x}) = 1 | \mathbf{x} \sim P_c\right) = \int_{\mathcal{R}_1} P_c(\mathbf{x}) d\mathbf{x} \\ P_{\mathrm{MD}} &= \Pr\left(F(\mathbf{x}) = 0 | \mathbf{x} \sim P_s\right) = 1 - \int_{\mathcal{R}_1} P_s(\mathbf{x}) d\mathbf{x}. \end{aligned}$$

The well known result of detection theory is that any detector must satisfy

$$(1 - P_{FA}) \log \frac{1 - P_{\mathrm{FA}}}{P_{\mathrm{MD}}} + P_{FA} \log \frac{P_{\mathrm{FA}}}{1 - P_{\mathrm{MD}}} \le D_{KL}(P_c || P_s), \tag{1.3.3}$$

which shows the importance of the KL divergence in the definition of steganographic security 1.1.1. If the Steganographic scheme is secure $(D_{\mathrm{KL}}(P_c||P_s) = 0)$, than the inequality 1.3.3 says that regardless the detector F Eve posses, it's performance is not better than random guessing.

A good steganographic detector needs to have low probability of false alarm. This is because in practice, the steganographic communication is usually repetitive, and images detected as stego are likely subjected to further forensic analysis with the goal to determine the steganographic program, the stego key, and eventually extract the secret message. Forensic analysis is usually computationally very expensive and time consuming, since it may use brute force dictionary attacks. Consequently, high false alarm rate of the detector F can overload Eve's forensic resources. Taking into account these practical circumstances, even the detector with very low probability of false alarm P_{FA} and relatively high missed detection rate $P_{\mathrm{MD}} \sim 0.5$ can still be very useful.

In practice, the prior probabilities of encountering a cover or stego image are unknown, which together with the requirement on the low probability of false positives, forces Eve to use almost exclusively the Neyman-Pearson setting. Eve's goal is to find a detector with the probability of false alarm bounded by the design parameter α, $P_{\mathrm{FA}} \le \alpha$, while maximizing the detection accuracy $P_{\mathrm{Acc}}(\alpha) = 1 - P_{\mathrm{MD}}(\alpha)$.

Given the bound on false alarms, α, the optimal Neyman-Pearson detector can be embodied by the likelihood ratio test (LRT):

$$\text{Decide } H_1 \text{ when } L(\mathbf{x}) = \frac{P_s(\mathbf{x})}{P_c(\mathbf{x})} > \gamma, \tag{1.3.4}$$

where $\gamma > 0$ is a threshold determined from the equation

$$\int_{\mathcal{R}_1} P_c(\mathbf{x}) d\mathbf{x} = \alpha,$$

and

$$\mathcal{R}_1 = \{\mathbf{x} \in \mathcal{C} | L(\mathbf{x}) > \gamma\}$$

is the critical region of the detector. The ratio $L(\mathbf{x})$ is called the likelihood ratio.

1.3.3. How to deal with high dimension. The approach to construct the optimal detector through likelihood ratio test 1.3.4 presented in the previous section is not practical for steganalysis of images. The dimension of the space of all images $\mathcal{C}$ is too big to obtain an accurate estimates of P_c and P_s. To escape from the construction of the detector F in space $\mathcal{C}$, $\mathcal{C}$ is frequently projected to a lower dimensional space $\mathcal{X}$ using a set of features $\mathbf{f}$. In steganography, features are usually real numbers and the space $\mathcal{X}$ is the Euclidean space $\mathcal{X} = \mathbb{R}^d$, where d is the dimension of the space. Assuming that $\mathcal{X} = \mathbb{R}^d$, the feature set can be represented by a mapping $\mathbf{f} : \mathcal{C} \mapsto \mathcal{X} = \mathbb{R}^d$, $\mathbf{f} = (f_1(\mathbf{x}), \ldots, f_d(\mathbf{x})) \in \mathbb{R}^d$, where each $f_i : \mathcal{C} \mapsto \mathbb{R}$. Random variables representing the cover $\mathsf{x} \sim P_c$ and the stego $\mathsf{y} \sim P_s$ are thus transformed into corresponding random variables $\mathbf{f}(\mathsf{x})$ and $\mathbf{f}(\mathsf{y})$ with probability distributions p_c ans p_s on $\mathbb{R}^d$. Obviously, the features have to be chosen so that the detection problems can be solved with the highest accuracy. Since the detection problems of targeted and blind steganalysis are different, the scope of the feature set is different as well.

Feature set for blind steganalysis should be constructed so that all steganographic schemes (including future schemes) irrespective to the embedding mechanism are detectable. Formally written, for any steganographic scheme S we want

$$D_{\mathrm{KL}}(P_s||P_c) > \epsilon \Rightarrow D_{\mathrm{KL}}(p_s||p_c) > 0.$$

Such a feature set is called *complete*. In practice, the requirement of completeness is relaxed to the weaker property, namely the requirement that it has to be hard to practically construct a stego scheme for which $D_{\mathrm{KL}}(p_s||p_c) = 0$. The dimensionality of the feature set for blind steganalysis is typically large, because for every steganographic scheme, there should be a feature with different values on cover and stego objects. The feature set for blind steganalysis can be understood as a low-dimensional model of cover images.

In contrast, feature set in targeted steganography can be much simpler. Even one feature may be enough to detect a specific steganographic scheme (if more than one feature is needed, than there has to exist a mapping that maps more features to one). Obviously, the feature set for targeted steganalysis has to be properly tailored to detect a given steganographic algorithm.

If we replace the probabilities P_c and P_s in the definition of steganographic security 1.1.1 by their lower dimensional counterparts p_c and p_s, the steganographic security will be defined *with respect to the feature set.*

1.3.4. Steganalysis as a Classification. Despite the substantial reduction of the dimensionality of the space, where the decision function is constructed, the probability density functions p_c and p_s on the projection space $\mathcal{X}$ can be rarely estimated in practice. This is because Eve does not have parametric models of probability distributions of cover p_c and stego p_s objects, the dimensionality of the projection space $\mathcal{X}$ is still too high obtain to accurate non-parametric models of p_c and p_s, and the set of examples of cover and stego objects is limited. The last two facts are especially relevant in blind steganalysis, where the dimensionality of the feature sets aspiring to be complete in the above practical sense is of the order of $10 - 10^3$.

The decision function $F : \mathcal{X} \mapsto \{0, 1\}$ is fully described by the critical region $\mathcal{R}_1$. To make the decision on sample $\mathbf{x}$, Eve does not need to know the probabilities $p_c(\mathbf{x})$ or $p_s(\mathbf{x})$. She just needs to know if $\mathbf{x} \in \mathcal{R}_1$. The probabilities $p_c(\mathbf{x})$ and $p_s(\mathbf{x})$ were needed only to identify the critical region $\mathcal{R}_1$ by means of the likelihood ratio test.

To escape from the need to estimate the conditional probabilities $p_c(\mathbf{x})$ or $p_s(\mathbf{x})$, the detection problems 1.3.1 and 1.3.2 are frequently formulated as classification problems, which are solved by algorithms from pattern recognition (machine learning) designed for problems with large dimension with relatively few number of examples[1].

To solve the simple hypothesis testing problem (1.3.2), Eve trains a classifier separating cover images from images with messages embedded by a specific steganographic algorithm. She does so by collecting sufficiently *large* and *diverse* database of cover objects, uses the steganographic algorithm to create stego objects containing messages with uniformly distributed lengths, and then trains a classifier of her choice on the database.

[1]This reformulation is a practical embodiment of Vapnik's famous motto "When solving a problem of interest, do not solve a more general problem as an intermediate step. Try to get the answer that you really need but not a more general one."

The composite hypothesis problem 1.3.1 in blind steganalysis is more difficult to solve due to the lack of knowledge about steganographic algorithm. Eve needs to create a classifier recognizing cover images in the feature space. Everything that does not look like a cover image is detected as a stego image. Properly written, Eve needs to identify the null-hypothesis region $\mathcal{R}_0$ containing the features of cover images. This problem is known in machine-learning as a novelty or anomaly detection problem. It aims to find the region $\mathcal{R}_0$ so that

$$p_c(\mathcal{R}_0) = \int_{\mathcal{R}_0} p(\mathbf{x})\mathrm{d}\mathbf{x} \geq 1 - \alpha,$$

where α is Eve's desired false positive rate (probability of detecting cover image as stego). This approach to steganalysis posseses an important advantage. As the classifier is trained solely on cover images, it does not have to be retrained when new embedding methods appear. The potential problem with this approach is that the database has to be very large and *diverse*. Especially the diverse adjective is important, because Eve does not want to identify cover images with unusual pre-processing (blurring, etc) as stego images.

The steganalyzers based on the combination of features and tools of pattern recognition are very flexible. Eve can always exchange the machine-learning engine, add new more powerful steganographic features, or simply extent the training database (all these actions require retraining of the classifier) to improve the detector. Thus, feature-based steganalysis benefits from the evolution if machine-learning and statistical modeling.

1.3.5. Influence of cover images on steganography and steganalysis. The choice of the cover image has a considerable influence on the security of the steganographic system. The choice of the cover image source should take into account the following factors:

- Size. In general, it is easier to detect messages of the same relative length in large images than in smaller ones. This is because shorter image provides smaller statistics, which is inherently more noisy than statistics calculated in longer images (see [**31**]).
- Number of color channels. If the cover image has more than one color channel, the attacker can utilize a possible correlation between color channels, which can substantially increase her chances. More color channels also provide larger statistics.
- Noise. Noisy images, such as scans of films or analog photographs are particularly good for the steganographer. High resolution scans can resolve the individual grains in the photographic material and this graininess manifests itself as high frequency noise. Since the noise is random by its nature, it is hard to model and the steganography hidden within noise of these images is very difficult to detect.
- Texture. More textured images are better for steganography than images with large smooth areas. The reasons are the same, as in the case of noise images. Textured images contain more high frequency components, which are more difficult to model.

This dissertation deals exclusively with JPEG images, where the effect of the cover work on the security is less pronounced, than in spatial domain steganography. This is because the effect of JPEG compression is similar to low-pass filtering, which effectively removes high frequency components that are hard to model.

1.4. Dissertation Goals and Outline

The main emphasis of the dissertation is on identification of the stego algorithm used to hide the message in a JPEG image. It is expected that this tool will be useful for law enforcement and forensic analysts, because identification of the stego program is the first necessary step towards extracting the secret message. The detector of the stego algorithm is developed under the condition of unknown compression history of the image, which vastly improves its applicability, because the conditions to some extent simulate the conditions expected in the real use. On the other hand, the very same conditions make the detection problem inherently difficult, since the statistics of JPEG images depends on the compression history. In order to compensate for the differences in image's statistics, instruments recovering the compression history of JPEG images, namely detector of double JPEG compression and estimator of primary quality factor, were developed (Chapter 3). The main advantage of both instruments with respect to prior art is that they can recover the compression history not only of cover but also stego images. Experiments also show that both tools also offer higher accuracy.

The detector of stego algorithms (further called *blind steganalyzer*), described in Chapter 5, recognizes one of 6 current popular steganographic algorithms for JPEG images (F5 [**72**], Model Based Steganography without [**56**] (MBS1) and with [**57**] deblocking (MBS2), JP Hide&Seek [2], OutGuess [**55**] ver. 0.2 with histogram correction, and Steghide [**27**]). It is based on feature extraction and supervised training of two banks of multi-classifiers implemented by Support Vector Machines. The first bank targeted to single-compressed images contains a separate multi-classifier for each JPEG quality factor from a range of 34 quality factors. The second bank consists of two multi-classifiers for double-compressed images for the same range of primary quality factors, as the first bank. Multi-classifier banks are preceded by a pre-classifier detecting double-compression and estimating the primary quantization table. The accuracy of the blind steganalyzer, examined under various conditions, is presented in Chapter 5. The feature set developed especially for the blind steganalyzer (described in Chapter 4) presents nowadays the state of art [**51**] stego features for JPEG images.

A universal steganalyzer is a steganalyzer capable of detecting any steganographic algorithm. Its most important property is the ability to detect images produced by algorithms with completely novel embedding mechanism as stego. Thus, the universal steganalyzer needs to be able to generalize to novel stego schemes. Chapter 6 examines various approaches to the design of a universal classifier and discusses their advantages and disadvantages.

With the increasing number of new steganographic algorithms, the issue of equitable comparison and verification of security of steganographic schemes is the most importance. Since the comparison cannot be carried directly in the space of all cover objects $\mathcal{C}$, a good benchmark should be dependent only on the model chosen to represent cover and stego objects (the feature set). Chapter 8 proposes Maximum Mean Discrepancy (MMD) as a measure of steganographic security. While the KL divergence would be preferable, because it is a more fundamental quantity, there are practical difficulties in computing it from data obtained from a test database of images. On the other hand, the MMD is well understood theoretically and numerically stable even in high-dimensional spaces, which makes it an excellent candidate for benchmarking in steganography.

[2]Can be obtained from: `http://linux01.gwdg.de/~alatham/stego.html`

Part 1

Blind steganalyzer for JPEG images

CHAPTER 2

Outline of the Blind Steganalyzer

2.1. Goals and design choices

The presented blind steganalyzer is targeted to JPEG images. The restriction to JPEG images enables to achieve higher accuracy, while it does not reduce the applicability too much, as JPEG format is the most ubiquitous image format in use today. The blind steganalyzer is designed to recognize 6 current popular steganographic programs: F5 [**72**], Model Based Steganography without [**56**] (MBS1) and with [**57**] deblocking (MBS2), JP Hide&Seek[1], OutGuess [**55**] ver. 0.2 with histogram correction, and Steghide [**27**]. The very first publicly available steganographic program Jsteg[2] and the state-of-the-art MMx algorithm [**37**] were intentionally left out, so that it will be possible to evaluate how well the blind steganalyzer handles stego images produced by previously unseen stego methods. The 6 stego programs were carefully selected to cover essentially all types of stego algorithms available today. The F5 uses a different embedding operation than LSB flipping and incorporates matrix embedding to minimize the number of embedding changes. MBS1 and MBS2 use LSB flipping and are designed to preserve a model of DCT coefficients. OutGuess and Steghide preserve the first order statistics. OutGuess does so by making additional changes (statistical restoration) while Steghide exchanges pairs of coefficients. JP Hide&Seek, due to Allan Latham is a heuristically improved modification of Jsteg.

All algorithms above, with the exception of OutGuess and F5, accept as input JPEG images and embed by directly manipulating their DCT coefficients. Thus, when a single-compressed image is used as cover, the stego image is never recompressed (double-compressed). On the other hand, implementation of F5 and OutGuess always decompress image into spatial domain, even when it is already in JPEG format. Prior to embedding a message, the decompressed image is JPEG compressed with either a default or user supplied quality factor (the default quality factor is 75 for OutGuess and 80 for F5). If the quantization matrix of input and output JPEG image is not the same, the resulting image is double-compressed (proper definition of double-compression is in Section 2.4). The probability that the stego-image produced by F5 or Outguess is double-compressed is high, because most users are not aware of the effects of different quality factors.

In order to reduce storage and computational resources needed to construct the blind steganalyzer (see more detailed discussion in Chapter 5), two simplifying assumptions were accepted. First it is assumed that the cover image is either a single-compressed JPEG image or an image that was never JPEG compressed (image is stored in some raw format). The second assumption is that double-compressed images produced by F5 and OutGuess algorithms have either quality factor 75 or 80.

There are several challenges that need to be overcome to create an accurate detector of steganography. First, the JPEG format accepts as a parameter the quantization table(s). The quantization tables drive the quantization of DCT coefficients and thus change their statistical properties. This effectively enlarges the space of covers and further complicates

[1] `http://linux01.gwdg.de/~alatham/stego.html`

[2] `http://zooid.org/~paul/crypto/jsteg/`

steganalysis because a classifier trained on one quality factor may give less accurate results on images with a different quality factor [**50**]. Second, multiple JPEG compression may dramatically change the statistics of DCT coefficients and thus cause some steganalysis methods to fail [**17**]. Problems with multiple JPEG compression can be solved by calculating features from the spatial domain of the images (JPEG image after decompression). Recent comparisons provided in [**51, 61, 74, 49**] show that the features computed directly from DCT coefficients, where the embedding changes are done, provide more accurate detection results.

2.2. Prior Art

The idea of using a collection of steganographic features together with machine learning tools is not new. The first known application used features computed from quality metrics [**3**] together with linear regression to detect the presence of watermark in the image. Linear regression was used not only to detect the watermark, but also to classify its type. Another interesting aspect of this work is that the feature extraction used the image together with its blurred version in order to calibrate the features. The same authors later proposed a different set of features based on binary similarity measures targeted to steganography [**4, 2**]. The linear regression was replaced with the more powerful Support Vector Machine classifier.

Farid [**16, 41**] constructed the features from wavelet and local angular harmonic decomposition of the image. Features calculated from the wavelet decomposition comprised of higher-order moments of distribution coefficients from several high-frequency sub-bands and higher-order moments of distribution of coefficient's local linear prediction errors. The calculation of features from local angular harmonic decomposition is more mathematically involved and we refer to original publication [**41**] for their description. Farid's work introduced the concept of one-class classification to steganography to construct universal steganalyzer (composite hypothesis test 1.3.1). The employed algorithm for one-class classification covers the cluster of features extracted from cover images by a pre-determined number (6) of multi-dimensional spheres. Everything that falls within a hypersphere is classified as cover, everything that falls outside is classified as stego image.

Several other feature sets for steganography in spatial domain were proposed, such as the center of gravity of the histogram characteristic function [**26**], absolute moments of the histogram characteristic function constructed in the wavelet domain [**74**], higher-order absolute moments of the image noise residual in the wavelet domain [**23**], and the statistics of full-frame DCT coefficients [**71**].

The steganalytic features for JPEG images should be calculated directly in the DCT domain, where the embedding changes occur. The first feature set targeted solely to steganalysis of JPEG images was introduced by Fridrich [**17**]. The calculation of features utilizes nowadays standard calibration procedure [**19**] to increase the sensitivity of features to embedding changes and decrease the sensitivity to image content. In [**49**], authors experimentally compared features calculated from DCT domain [**17**] with features calculated from spatial domain [**41**] on selected steganographic algorithms for JPEG images and concluded that features extracted from DCT domain offer superior performance. Shi et al. [**61**] proposed features based on Markov models of dependencies of absolute values of DCT coefficients. The calculation of features does not use calibration, because according to authors, calibration increases the sensitivity of feature sets to the JPEG quality factor and their intention was to create a feature set insensitive to the quality factor.

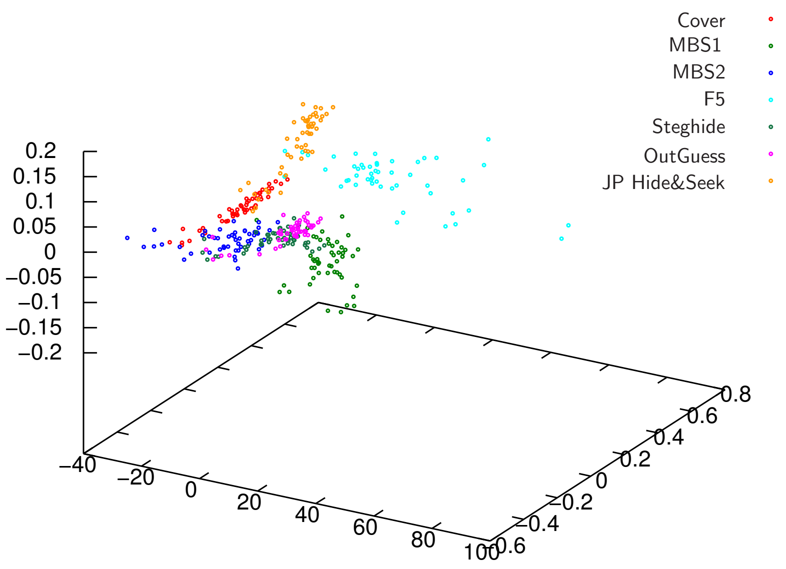


FIGURE 2.1. Feature space spanned by eigenvectors of the three largest eigenvalues after principle component transformation of Merged features (Chapter 4). Features are extracted from cover images and images fully embedded by F5, JP Hide&Seek, MBS1, MBS2, OutGuess and Steghide.

2.3. Outline of the proposed Blind Steganalyzer

The combination of models of cover images in low-dimensional space combined with classification tools of machine learning proved to be well suited for the detection of the embedding algorithm [**3, 49**]. Images embedded using different steganographic algorithms form distinct clusters in the feature space, as can be seen on Figure 2.1, because different steganographic algorithms introduce different artifacts into images, which shifts the feature vector along different directions in the feature space. For instance, the F5 algorithm [**72**] increases the number of zeros and slightly decreases the number of all non-zero DCT coefficients. By using a large number of features, the chances that two different embedding programs will produce feature vectors located in different parts of the feature space increases. This exact property is important for classification of steganographic algorithms.

The presented Blind Steganalyzer uses 274 Merged features computed directly from quantized DCT coefficients (described in detail in Chapter 4). All features are calibrated by taking differences between quantities calculated from the stego image and from an estimate of the cover image. Calibration makes features more sensitive to embedding changes and relatively insensitive to image content. Its disadvantage is that it makes features more sensitive to the quality factor of the stego image and the combinations of quality factors in double-compressed images [**19**]. This issue is resolved by first detecting selected cases of double-compressed images and then sending the stego image to an appropriate multi-classifier, depending on the stego image quality factor and the fact whether or not it was double-compressed. The main reason for choosing Merged feature set is their excellent performance documented by comparisons to prior art (Section 4.5).

Figure 2.1 shows a high-level description of the proposed steganalyzer. Its structure logically follows from the assumptions made in Section 2.1. The steganalyzer consists of two multi-classifiers, one for single-compressed and one for double-compressed images. Multi-classifiers are preceded by a detector of double-compression, which pre-classifies the image under inspection and then sends it to the appropriate multi-classifier.

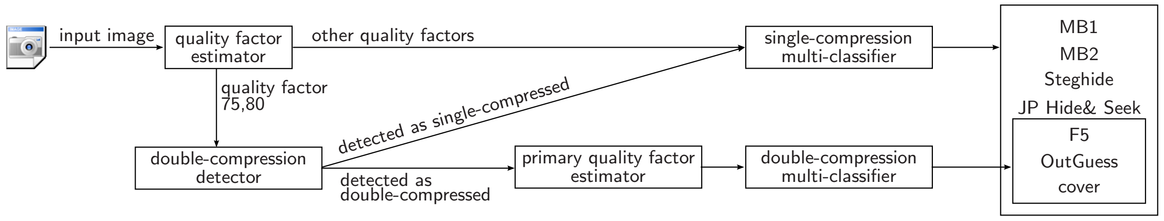

input image
quality factor estimator
other quality factors
single-compression multi-classifier
quality factor 75,80
detected as single-compressed
double-compression detector
detected as double-compressed
primary quality factor estimator
double-compression multi-classifier
MB1
MB2
Steghide
JP Hide& Seek
F5
OutGuess
cover

FIGURE 2.1. Outline of the blind steganalyzer for JPEG images.

The single-compression multi-classifier is, in fact, a collection of 34 multi-classifiers $\mathcal{S}_i$, indexed by the quality factor $i \in \mathcal{Q}_{34}$,

$$\mathcal{Q}_{34} = \{63, 64, \ldots, 93, 94, 96, 98\}.$$

This set of quality factors was a compromise between the desire to create as general classifier as possible and the limited computational and storage resources. Quality factors 95 and 97 were omitted because there is not a significant difference between quantization matrices Q_{95} and Q_{97} and quantization matrices Q_{96} and Q_{98}. Each multi-classifier $\mathcal{S}_i$, $i \in \mathcal{Q}_{34}$, is trained to assign JPEG images with quality factor i to 7 categories—covers and 6 stego algorithms described above.

The double-compression multi-classifier consists of two multi-classifiers $\mathcal{D}_{75}$ and $\mathcal{D}_{80}$ that assign double-compressed JPEG images with secondary quality factors 75 and 80 to cover, F5, or OutGuess, because only these two algorithms can create double-compressed images during embedding under the assumptions made earlier. An integral part of the multi-classifier is a primary quantization matrix estimator, which provides the primary quantization matrix for calibration of double-compressed images.

The whole process of analyzing an image starts by inspecting its quality factor[3] i. If $i \notin \{75, 80\}$, the image is sent directly to the *multi-classifier for single-compressed images* $\mathcal{S}_i$. If $i \in \{75, 80\}$, the image is forwarded to the *double-compression detector*. If the double-compression detector detects the image as single-compressed, the image is sent to $\mathcal{S}_i$. If the image is detected as double-compressed, the primary quality factor of the image is estimated, added as an additional 275-th feature, and sent with the image to the *multi-classifier for double-compressed images* $\mathcal{D}_i$, which classifies it as either cover, F5, or OutGuess.

The accuracy of the double-compression detector has a major impact on the accuracy of classification of images with quality factors 75 and 80. If the double-compression detector deems a single-compressed image as double-compressed, it can be classified only as cover or embedded by F5/OutGuess, even though the image was embedded by a different algorithm. Thus, the double-compression detector has to be tuned to a low false positive rate (incorrectly detecting a single-compressed image as double-compressed).

The following part of the dissertation describing the construction of the Blind Steganalyzer is organized as follows. Chapter 3 describes the details of double-compression detector and primary quantization matrix estimator and compares them to prior art. Chapter 4 describes the Merged feature set used by the blind steganalyzer. The same chapter explains the calibration process and shows how the calibration increases the sensitivity of features to embedding. Chapter 5 describes the multi-classifier for single-compressed and double-compressed images and presents error rates of the complete blind steganalyzer on testing database. Since the constructed blind steganalyzer does not generalize to novel embedding schemes well, Chapter 6 explores several solutions to construct a universal steganalyzer.

[3]If the image has a non-standard quantization table, the closest standard quantization table (and thus a quality factor) is found by matching low-frequency quantization steps (see Section 3.2.1).

Chapter 7 is devoted to an alternative use of the combination of machine learning algorithms and feature sets in steganography and steganalysis.

In the rest of this chapter, basics of JPEG compression are described in Section 2.4, and the database of images created for experiments in this part of dissertation is described in Section 2.5.

2.4. Basics of JPEG Compression

The JPEG format, approved as an ISO standard no. 10918-1 in 1994, is the most commonly used image format today. Although more sophisticated version JPEG 2000 based on vawelet transformation was approved as a replacement, original JPEG is still favored over JPEG 2000. This section briefly recapitulates the basic properties of the JPEG format that are relevant to problems solved in this dissertation. A detailed description of the format can be found in [**47**][4].

2.4.1. JPEG compression and decompression. During JPEG compression, the image is first divided into disjoint 8×8 pixel blocks $B_{rs},\ r, s \in \{0, \dots, 7\}$ (r, s are indeces of coefficients within an 8×8 block). Each block is transformed using the Discrete Cosine Transformation (DCT)

$$d_{ij} = \frac{w(i)w(j)}{4} \sum_{r,s=0}^{7} \cos\frac{\pi}{16} i(2r+1) \cos\frac{\pi}{16} j(2s+1) B_{rs},\ i, j \in \{0, \dots, 7\}$$

where $w(0) = \frac{1}{\sqrt{2}}$ and $w(r > 0) = 1$. The DCT coefficients d_{ij} are then divided by quantization steps stored in the quantization matrix Q_{ij} and rounded to integers

$$D_{ij} = \text{round}\left(\frac{d_{ij}}{Q_{ij}}\right),\ i, j \in \{0, \dots, 7\}. \tag{2.4.1}$$

The i, j-th DCT coefficient in the k-th block is denoted as $D_{ij}(k),\ k \in \{1, \dots, n_b\}$, where n_b is the number of all 8×8 blocks in the image. The pair $(i, j) \in \{0, \dots, 7\} \times \{0, \dots, 7\}$ is called the *spatial frequency* (or *mode*) of the DCT coefficient. The JPEG compression finishes by ordering the quantized coefficients D_{ij} along a zig-zag path, encoding them, and finally applies loss-less compression.

The decompression works in the opposite order. After reading the quantized DCT blocks from the JPEG file, each block of quantized DCT coefficients D is multiplied by the quantization matrix Q, $\hat{d}_{ij} = Q_{ij} \cdot D_{ij}$, and the Inverse Discrete Cosine Transformation (IDCT) is applied to $\hat{d}_{ij}$,

$$\hat{b}_{rs} = \sum_{i,j=0}^{7} \frac{w(i)w(j)}{4} \cos\frac{\pi}{16} i(2r+1) \cos\frac{\pi}{16} j(2s+1) \hat{d}_{ij}$$

Due to the rounding and quantization step (2.4.1) in the compression, the values $\hat{b}_{rs}$ from IDCT are not integers and may lie outside the allowed range for pixel values (usually $[0, 255]$ for 8 bit images). To resolve this, the values $\hat{b}_{rs}$ are rounded and truncated to the admissible range. For the decompressed pixel values $\hat{B}$ holds

$$\hat{B} = \text{trunc}(\text{round}(\text{IDCT}(Q \odot D))), \tag{2.4.2}$$

where $\odot$ denote element-wise matrix multiplication. Due to repeated rounding and truncation involved in compression and decompression, $\hat{B}$ will in general differ from the original block B.

[4]Good overview of JPEG compression standard can be also found on the wikipedia `http://en.wikipedia.org/wiki/JPEG`.

JPEG image is called *double-compressed* if the JPEG compression was applied twice, each time with a different quantization matrix and with the same alignment with respect to the 8×8 grid. The first matrix Q^1 is called the *primary quantization matrix* and the second matrix Q^2 is called the *secondary quantization matrix*. Additionally, a specific DCT *coefficient* D_{ij} is pronounced as double-compressed if and only if $Q^1_{ij} \neq Q^2_{ij}$.

Note that the definition of double-compression does not cover the case when the image is spatially shifted (cropped) after decompression but prior to the second compression. These images are not considered as double-compressed in this dissertation, even though the previous compression will undoubtedly have some effect on steganalysis.

Color components of the image are compressed independently to each other by the algorithm described above. Although it is not mandatory, most implementations of JPEG compression transform color space of the image to YCbCr representation. Luminance component Y is compressed in full resolution, while the color components Cb and Cr are frequently downsampled by a factor of 2. Color components Cb and Cr are also quantized with a coarser quantization matrix (2.4.3) than the luminance component Y (2.4.4). The reason for the different processing of color and luminance components is that the human eye is more sensitive to changes in luminance than to changes in color. This difference in human's eye perception allows higher compression of color components without visually degrading the image (for humans).

2.4.2. Quantization Matrices. Although the JPEG standard does not specify quantization matrices, most implementations of JPEG compression use a set of quantization matrices used in the reference implementation[5] provided by the Independent JPEG Group. These matrices were selected to give good visual results, yet keeping good compression ratio. Since these matrices became de facto part of the standard, we refer to them as *standard* matrices and describe them in the rest of this section.

The standard quantization matrices are indexed by a quality factor $q \in \{1, \ldots, 100\}$. The individual quantization steps for quality factor q are calculated according to the following formula

$$Q_{ij}(q) = \max\left\{1, \mathrm{round}\left(s(q) \cdot Q_{ij}(50)\right)\right\},\ i, j \in \{0, \ldots, 7\},$$

where

$$s(q) = \begin{cases} 2 - \frac{2q}{100} & \text{if } q > 50, \\ \frac{50}{q} & \text{otherwise.} \end{cases}$$

Since the scaling factor $s(50) = 1$, the quantization matrix $Q(50)$ for quality factor 50 is used to define quantization matrices for other quality factors. Quantization matrix $Q_{\mathrm{LUM}}(50)$

$$Q_{\mathrm{LUM}}(50) = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 93 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}, \tag{2.4.3}$$

[5]`ftp://ftp.simtel.net/pub/simtelnet/msdos/graphics/jpegsr6.zip` The algorithm for calculating the quantization matrices is implemented in the function "jpeg_quality_scaling" in the file "jcparam.c"

serves as a pre-image of quantization matrices for luminance component, quantization matrix $Q_{\text{Color}}(50)$

$$Q_{\text{Color}}(50) = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix} \tag{2.4.4}$$

serves as a pre-image of quantization matrices for color components of the image.

2.5. Database of Images

The database of images used for experiments in this part of the dissertation was created from 6006 images of natural scenes. Images were taken under varying conditions (exterior and interior images, images taken with and without flash and at various ambient temperatures) with the following digital cameras: Nikon D100, Canon G2, Olympus Camedia 765, Kodak DC 290, Canon PowerShot S40, images from Nikon D100 downsampled by a factor of 2.9 and 3.76, Sigma SD9, Canon EOS D30, Canon EOS D60, Canon PowerShot G3, Canon PowerShot G5, Canon PowerShot Pro 90IS, Canon PowerShot S100, Canon PowerShot S50, Nikon CoolPix 5700, Nikon CoolPix 990, Nikon CoolPix SQ, Nikon D10, Nikon D1X, Sony CyberShot DSC F505V, Sony CyberShot DSC F707V, Sony CyberShot DSC S75, and Sony CyberShot DSC S85. All images were taken either in the raw TIFF format or in a proprietary manufacturer raw data format, such as NEF (Nikon) or CRW (Canon) converted to the 24-bit TIFF format. The image resolution ranged from 800×631 for the scaled images to 3008×2000. Scaled images were included in order to increase the diversity of the database. Since all images were stored in the raw format, the compression history of the images was known.

Before conducting any experiments, images were divided into two disjoint groups. The first group consisting of 3500 images from the first 7 cameras on the list (including the down-sampled images) was used to create the training examples. The second group with remaining 2506 images was used to create testing images. Training and testing sets were processed in the same way. The testing subset contains images taken by different cameras and photographers, than images in the training set. The different origin of images in the testing set together with their strict separation increases the credibility of errors calculated on the testing set.

The database contained single-compressed stego images with 34 different quality factors from the set

$$\mathcal{Q}_{34} = \{63, 64, \ldots, 93, 94, 96, 98\},$$

which is the set of quality factors the steganalyzer is designed to detect. Images were embedded by 6 steganographic algorithms: F5, MBS1, MBS2, JP Hide&Seek, OutGuess, and Steghide. The length of the messages embedded by all algorithms except MBS2 was 100%, 50%, and 25% of the embedding capacity for each algorithm. All MBS2 images were embedded only with 30% of the capacity of MBS1 because during embedding of longer messages the deblocking part of MBS2 often fails. The capacity of JP Hide&Seek was estimated as 10% of the size of the JPEG file, as recommended by its author. The implementation of OutGuess was modified to produce images with quality factor smaller than 75.

The double-compressed stego images were created by OutGuess and F5. Again, the length of embedded messages was 100%, 50%, and 25% of embedding capacity for each algorithm and image. The double-compressed images were prepared with 34 different primary quality factors $\mathcal{Q}_{34}$ and with two different secondary quality factors: 75, which is the default quality factor of OutGuess, and 80, the default quality factor of F5.

The database contained single-compressed cover images with quality factors $\mathcal{Q}_{34}$ and double-compressed cover images with primary quality factors $\mathcal{Q}_{34}$ and secondary quality factors 75 and 80. The total number of images in the database was $|\mathcal{Q}_{34}| \times 17 \times 6006 + |\mathcal{Q}_{34}| \times 2 \times 7 \times 6006 \approx 6,330,000$.

CHAPTER 3

Detection of double-compression

3.1. Motivation

This chapter addresses the problems of detection of double-compression in JPEG images and estimation of primary quantization matrix under the assumption that the image can potentially contain an embedded message. Even though the first problem can be understood as a sub-problem of the second one, two separate tools for each problem were created. The double-compression detector (DC detector) can exploit statistics from all DCT coefficients in the image, while the primary quantization steps detector, which is an intermediate step in the estimation ofthe primary quantization matrix, can utilize statistics of DCT coefficients on a single DCT mode only. The larger statistics exploited by the DC detector results in its higher accuracy, which is crucial, because it affects the accuracy of whole blind steganalyzer, as explained in Section 2.3.

Recall that a JPEG image is called double-compressed, if it was compressed twice, each time with a different quantization matrix. The quantization matrix used in the first compression is called the primary quantization matrix Q^1, the quantization matrix used in subsequent (second) compression is called the secondary quantization matrix Q^2 (JPEG compression was described in Section 2.4). Since the previous (primary) quantization matrix is not needed to decompress the JPEG image to the spatial domain, the JPEG image file does not keep track of the compression history of the image and only the latest (secondary) quantization matrix is stored within the file. The lost primary quantization matrix needs to be recovered for correct calibration of double-compressed images.

Recovering the compression history of the JPEG image is important not only for steganalysis, which is our major concern here, but also for forensic analysis. The fact that the image was double-compressed indicates manipulation. By determining double-compression history in smaller regions, we may discover inconsistencies caused by tampering the image. For example, when pasting an object into a decompressed JPEG and re-saving with a different JPEG quality factor, the pasted object may exhibit different JPEG compression artifacts than the rest of the image.

3.1.1. Effect of double-compression on DCT histograms. The effect of double-compression on the value of DCT coefficient D_{ij} can be modeled as

$$D_{ij} = \left[\left[\frac{d_{ij}}{Q^1_{ij}} \right] \cdot \frac{Q^1_{ij}}{Q^2_{ij}} \right], \tag{3.1.1}$$

where truncation (2.4.2) in the intermediate decompression step was omitted. The simplified model (3.1.1) unfolds the dependency of the double-compressed DCT coefficient D_{ij} on the combination of quantization steps Q^1_{ij} and Q^2_{ij}. The effects of double-compression on D_{ij} are best described by histograms of DCT coefficients

$$h_{ij}(m) = \sum_{k=1}^{n_b} \delta\left(|D_{ij}(k)| - m \cdot Q^2_{ij} \right), \tag{3.1.2}$$

where $k \in \{1, \ldots, n_b\}$ indexes the 8×8 blocks and δ is the indicator function, $\delta(x) = 1$ if $x = 0$ and $\delta(x) = 0$ when $x \neq 0$. We recognize two distinct artifacts of double-compression on the shape of the histogram $h_{ij}(m)$.

A *zero* occurs if there exists an integer $u > 0$ such that $Q^1_{ij} = u \cdot Q^2_{ij}$ (the primary quantization is coarser than the secondary quantization). In this case, $D_{ij} \approx u \cdot \left[\frac{d_{ij}}{Q^1_{ij}}\right]$ gives a restriction on the values of D_{ij}, which can now only attain values from the set $\{0, u, 2u, 3u, \ldots\}$. Thus, $h_{ij}(m) = 0$ for $m \in \mathbb{N} \setminus \{0, u, 2u, 3u, \ldots\}$. Equivalently,

$$h_{ij}(m) \begin{cases} > 0 & m \in \{0, k, 2k, 3k, \ldots\} \\ = 0 & m \bmod u \neq 0. \end{cases}$$

Figure 3.1(b) shows an example of a histogram exhibiting zeros at $m \in \{1, 3, 5, 7, \ldots\}$ ($Q^1_{ij} = 8$ and $Q^2_{ij} = 4$).

A *double peak* occurs when there exist integers $u, v \geq 0$ such that

$$uQ^1_{ij} = \frac{1}{2}\left((v-1)Q^2_{ij} + vQ^2_{ij}\right)$$

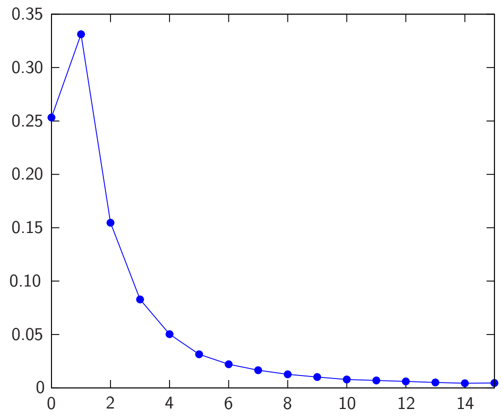
(the multiple $\left[\frac{d_{ij}}{Q^1_{ij}}\right] \cdot Q^1_{ij}$ falls in the middle of two multiples of Q^2_{ij} and no other multiple of Q^2_{ij} is closer). In this case, the multiple $\left[\frac{d_{ij}}{Q^1_{ij}}\right] \cdot Q^1_{ij}$ contributes approximately the same to both $(v-1)Q^2_{ij}$ and vQ^2_{ij}. Figures 3.1(c,d) show examples of double-peaks occurring at multiples $v = 2, 5, 8, \ldots$. A more detailed description of the impact of double-compression on the DCT histogram can be found in [**53, 40, 54**].

There is one more case deserving to be described, even though it does not have any significant effect on the histogram (3.1.2). *Divisor* occurs, when there exists an integer $u > 0$ such that $Q^2_{ij} = u \cdot Q^1_{ij}$ (the primary quantization is finer than the secondary quantization, a complementary case to zeros). The shape of the histogram (Figure 3.1(e)) is almost identical to the histogram of single-compressed DCT coefficients quantized by Q^2_{ij} (Figure 3.1(a)). Despite the quantization steps Q^1_{ij} and Q^2_{ij} being different, the DCT coefficients are not technically double-compressed, which makes the quantization step Q^1_{ij} undetectable.

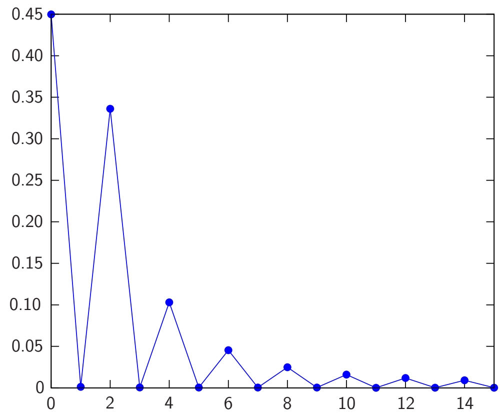
While the histogram $h_{ij}(m)$ of single-compressed DCT coefficients can be well-modeled by generalized Gaussian distribution [**30**], histograms of double-compressed DCT coefficient do not follow any known distribution. Figures 3.1(a-d) show histograms of DCT coefficients for a fixed mode $(0, 1)$ calculated from images sharing the same original raw image and the same secondary quantization matrix Q^2. We can see how different value of the primary quantization step Q^1_{01} produces histogram with completely different shapes. These substantial differences in histograms suggest that histograms (3.1.2) might be good features to detect primary quantization steps Q^1_{ij}.

3.1.2. Prior Art. To the best of our knowledge, the first work dedicated to the problem of estimation of the primary quantization matrix in double-compressed images is due Lukáš et al. [**40**][1]. Instead of restoring the whole primary quantization matrix, the authors focused on estimation of the primary quantization steps Q^1_{ij} for low-frequency DCT coefficients $(i, j) \in \{(0, 1), (1, 1), (1, 0)\}$, because the estimates for higher frequencies become progressively less reliable due to insufficient statistics. The publication discussed three

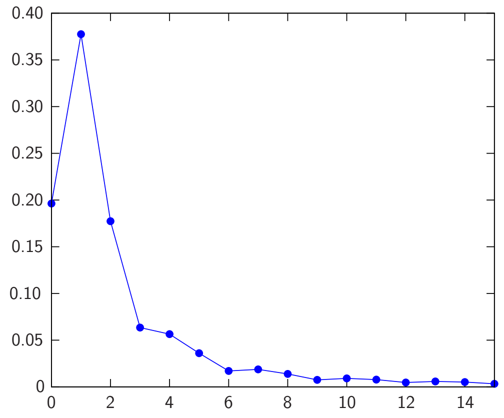
[1]The problem of detection of previous (single) JPEG compression from bitmap images was also investigated in [**15**].



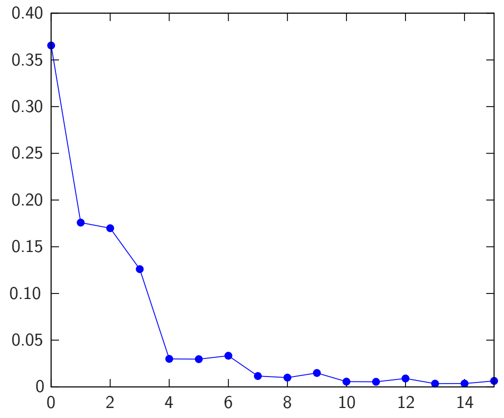
(a) $Q^1_{ij} = 4, Q^2_{ij} = 4$: histogram of a single-compressed DCT coefficient



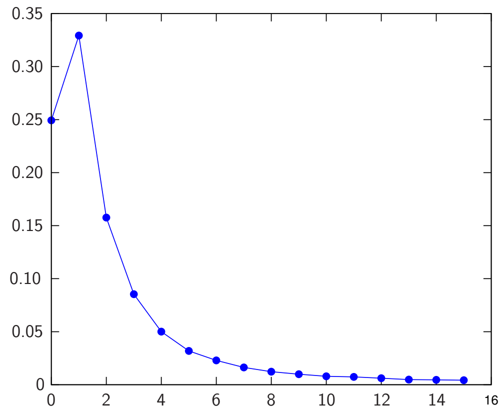
(b) $Q^1_{ij} = 8, Q^2_{ij} = 4$: histogram with zeros at multiples $1, 3, 5, \ldots$



(c) $Q^1_{ij} = 3, Q^2_{ij} = 4$: histogram with double peaks at multiples $(1, 2), (4, 5), (7, 8) \ldots$



(d) $Q^1_{ij} = 6, Q^2_{ij} = 4$: histogram with double peaks at multiples $(1, 2), (4, 5), (7, 8), \ldots$



(e) $Q^1_{ij} = 2, Q^2_{ij} = 4$: histogram of divisor.

FIGURE 3.1. Effect of double-compression on histograms of absolute values of DCT coefficients on a fixed mode $(0, 1)$. The secondary quantization step is in all four cases the same, $Q^2_{ij} = 4$, only the primary quantization step Q^1_{ij} varies. Figure (a) shows histogram of single-compressed DCT coefficients, Figure (b) shows histogram exhibiting zeros, Figures (c) and (d) show histogram exhibiting double-peaks, and, finally, Figure (e) shows histogram exhibiting divisor effect — histogram is not influenced by double-compression.

approaches. Two of them were based on matching the histograms of individual DCT coefficients of the inspected image to the histograms calculated from estimates obtained by calibration [**19, 17**] followed by simulated double-compression. Both histogram matching approaches were outperformed by the third method that used a collection of neural networks. One neural network was trained for each value of the secondary quantization step of interest, $Q^2_{ij} \in \{1, \ldots, 9\}$, to recognize the primary quantization step Q^1_{ij} in the range $[2, 9]$, for $Q^2_{ij} \in \{2, \ldots, 9\}$, and in the range $[1, 9]$, for $Q^2_{ij} = 1$. All neural networks used the same input feature vector

$$x = \{h_{ij}(2), h_{ij}(3), \ldots, h_{ij}(15)\}, \tag{3.1.3}$$

where $h_{ij}(m)$ denotes the histogram (3.1.2) described in Section (3.1.1). Interestingly, the feature vector (3.1.3) does not include $h(0)$ and $h(1)$. The reported accuracy of this neural network detector on cover JPEG images was better than 99% for estimation of low frequency quantization steps with frequencies $(i, j) \in \{(0, 1), (1, 1), (1, 0)\}$, and better than 95% for quantization steps for frequencies $(i, j) \in \{(2, 0), (2, 1), (1, 2), (0, 2)\}$. This Neural Network detector is compared to the presented solution in Section 3.3.2.

Shi et al. [**61**] proposed an idea for recovery of compression history of images based on the observation that the distribution of the first (leading) digit[2] of DCT coefficients in single-compressed JPEG images of natural scenes follows the generalized Benford distribution

$$p(x) = N \cdot \log\left(1 + \frac{1}{s + x^q}\right) \; x \in \{1, \ldots, 9\}, \tag{3.1.4}$$

where q and s are free parameters and N is a normalization constant. This fact is used to estimate the quantization matrix Q of images previously JPEG compressed but currently stored in some other lossless image format (such as TIFF or PNG) in the following manner. The inspected image is compressed with different quality factors Q_t. When $Q_t \neq Q$, the image will be double-compressed and the generalized Benford law (3.1.4) will be violated. On the other hand, when $Q_t = Q$, the first digit of DCT coefficients will follow the generalized Benford distribution for some q and s, because the statistics of DCT coefficients is not affected by double-compression. The publication also proposes to use the histogram of the first digit of DCT coefficients as a feature set (further called the Benford feature set) for detection of double-compressed images. Benford feature set is compared to the features set based on histograms $h_{ij}(m)$ in Section 3.3.1.1, by training and evaluating the classifiers on exactly the same database of images.

A completely different approach proposed in [**53, 54**] targets only detection of double-compression. Authors recognized that the double-compression artifacts in the histograms of DCT coefficients $h_{ij}(m)$ have periodic character. The periodicities caused by double-compression manifest themselves as detectable peaks in the Fourier transform of $h_{ij}(m)$. The reported detection accuracy (double-compressed image detected as double-compressed) of the method, estimated on 100 cover images, was most of the time 100% with 0% false alarm rate (single-compressed image detected as double-compressed). Unfortunately, we were unable to obtain the implementation from the authors and our own implementation produced results incompatible with those reported in [**54**], which prevented us from comparing the approaches.

There is a theoretical method for detection of primary quantization steps, which was not to the best of our knowledge published, but deserves to be mentioned. As the single-compressed DCT coefficients can be modeled by generalized Gaussian distribution, and the

[2]The first digit is understood as the first *valid* digit in the decimal representation of the number. The numbers are *not* padded by zeros to have the same number of digits. For example 2 and 25 have the same first digit 2.

effect of double-compression on DCT coefficients is known (3.1.1), parametrized distribution of double-compressed DCT coefficients for a fixed DCT mode can be derived. This parametrized distribution can be used to estimate Q^1_{ij} together with the nuisance parameters of the generalized Gaussian using maximum likelihood estimation. A potential flaw of this approach is that it does not take into account the effects of embedding changes on the statistics of DCT coefficients, which can be quite complex and consequently hard to model (if possible at all in blind steganalysis setting). Since the stego images are our primer concern, this interesting idea is not pursued further, despite its mathematical elegance.

All methods related to the problem of recovering compression history of JPEG images targeted cover images only. This is clearly not enough, since as was already mentioned, steganalysis benefits from the knowledge of the compression history of the stego image, whose statistics may be disturbed by embedding. Thus, special tools developed for images with possible stego content are needed.

3.2. The Proposed Approach

The differences in the shape of histograms of DCT coefficients of single-compressed and double-compressed images (Figure 3.1) suggest that the histograms can be used as features for a classifier. The good performance of the neural network approach reported in [**40**] supports this suggestion.

The careful examination of standard quantization matrices (2.4.2) for most quality factors reveals that DCT coefficients for higher modes are quantized with coarse quantization steps, making them almost all zeros. This makes the statistics of DCT coefficients on these high frequency modes insufficient for detection of double-compression and estimation of corresponding primary quantization steps. Therefore, both tools described here use statistics only from following set of 9 modes

$$\mathcal{L} = \{(1,0),(2,0),(3,0),(0,1),(1,1),(2,1),(0,2),(1,2),(0,3)\}\,. \tag{3.2.1}$$

3.2.1. Quality factor estimator. The JPEG standard allows using arbitrary quantization matrices that do not necessarily have to correspond to any standard quantization matrix for any quality factor. Because the multi-classifiers from which the blind steganalyzer is constructed are indexed by the quality factor, if the secondary JPEG quantization matrix Q^2 is non-standard, the closest standard quantization matrix $Q \in \mathcal{T}$ is found so that the image can be sent to the proper multi-class steganalyzer. This is achieved using the following formula

$$Q = \arg\min_{Q\in\mathcal{T}} \sum_{(i,j)\in\mathcal{L}} |Q_{ij} - Q^2_{ij}|,$$

where the sum is taken over a selected band of spatial frequencies $\mathcal{L}$. Since most nonzero DCT coefficients in natural images are in the low-frequency band, it is important to match these modes rather than modes corresponding to high spatial frequencies. The blind steganalyzer uses the band $\mathcal{L}$ for the range of quality factors from $\mathcal{Q}_{34}$.

3.2.2. Double-compression detector. The double-compression detector (DC detector) is an algorithm classifying images into two classes — single-compressed images and double-compressed images. It is implemented by a soft-margin Support Vector Machine ($C-$SVM, described in Appendix A) with Gaussian kernel $k(x,y) = \exp(-\gamma\|x-y\|^2)$. The feature vector is derived from histograms $h_{ij}(m), m \in \{0,\ldots,15\}$ (3.1.2) of absolute values of DCT coefficients $D_{ij}(k)$ in the inspected JPEG image. The histograms in the feature vector are calculated for the low-frequency DCT modes $\mathcal{L}$. The feature vector $\mathbf{x}$ can be formally written as

$$\mathbf{x} = \left\{ \frac{1}{C_{ij}} \left(h_{ij}(0), h_{ij}(1), \ldots, h_{ij}(15)\right) \middle| \; (i,j) \in \mathcal{L} \right\},$$

where C_{ij} are normalization constants $\left(C_{ij} = \sum_{m=0}^{15} h_{ij}(m)\right)$. The dimension of the feature vector x is $16 \times |\mathcal{L}| = 144$.

Even though the proposed approach is applicable to JPEG images with any secondary quality factor, because of the assumptions under which the blind steganalyzer is developed the double-compression detector was implemented only for two secondary quality factors 75 and 80. Double-compression detector uses a separate C-SVM dedicated to each secondary quality factor (SQF). This design with separated SVMs for each quality factor allows faster training on larger number of examples. The complexity of training of SVM is approximately $O(N^3)$, where N is the number of training examples. Consequently, to train one SVM on $2N$ examples is 8-times slower than to train 2 SVMs on N examples.

A noteworthy advantage of double-compression detector implemented by a single binary classifier is the possibility to shift the bias of the detector towards a lower false positive rate (detecting single-compressed JPEG image as double-compressed). Low false positive rate is important for the blind steganalyzer, as explained in Section 2.3.

3.2.3. Primary quality factor estimator. Due to the problem with insufficient statistics for high-frequency DCT modes, the complete quantization matrix cannot be restored without additional side information. This issue is alleviated by restricting the detection only to standard quantization matrices (defined in Section 2.4.2).

The primary quality factor estimator (PQF estimator) is divided into two parts. The first part detects primary quantization steps for 9 lowest AC modes $\mathcal{L}$, the second part finds the closest standard quantization matrix in the maximum likelihood sense.

3.2.3.1. *Detector of primary quantization steps.* The detector of the primary quantization steps is implemented as a collection of SVM-based multi-classifiers $\mathcal{F}_{Q^2_{ij}}$ indexed by the value of the secondary quantization step Q^2_{ij}. The standard quantization matrices for quality factors 75 and 80 have only 5 different quantization steps $\{4, 5, 6, 7, 8\}$ for all modes $(i,j) \in \mathcal{L}$. Thus, 5 multi-classifiers need to be constructed. Each multi-classifier $\mathcal{F}_{Q^2_{ij}}$ classifies into $n_{Q^2_{ij}}$ classes, where $n_{Q^2_{ij}}$ is the number of all possible values of the primary quantization step when the secondary quantization step is Q^2_{ij}. Theoretically, $n_{Q^2_{ij}} = 255$ since the quantization step can be represented by an 8-bit number and it cannot be zero. However in practice, the number of detected primary quantization steps is considerably smaller, because only steps from 34 quantization matrices with quality factors from

$$\mathcal{Q}_{34} = \{63, 64, \ldots, 93, 94, 96, 98\}$$

need to be estimated. This set was determined by the database of available JPEG images described in Section 2.5. The primary quantization steps detected for each secondary quantization step are shown in Table 3.1. The number of binary SVMs in one multi-classifier is $\binom{n_{Q^1_{ij}}}{2}$, because employed "max-wins" multi-classifier (see Appendix A.5) need one binary SVM for each pair of classes.

The feature vector x for the multi-classifier $\mathcal{F}_{Q^2_{ij}}$ is again formed by the histogram of absolute values of the first 16 multiples of Q^2_{ij} of all DCT coefficients $|D_{ij}(k)|$ for all blocks

SQS	Detectable PQS	#SVMs
4	$\mathcal{S}_4 = \{3, 4, 5, 6, 7, 8\}$	15
5	$\mathcal{S}_5 = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$	36
6	$\mathcal{S}_6 = \{4, 5, 6, 7, 8, 9, 10, 11, 12\}$	36
7	$\mathcal{S}_7 = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$	36
8	$\mathcal{S}_8 = \{3, 5, 6, 7, 8, 9, 10, 11, 12\}$	36

Table 3.1. Primary quantization steps (PQS) detectable by the multi-classifier for a given secondary quantization step (SQS). The last column (#SVMs) shows the number of binary Support Vector Machines in the multi-classifier.

$k = 1, \ldots, l$

$$x = \frac{1}{C_{ij}}(h_{ij}(0), h_{ij}(1), \ldots, h_{ij}(15)), \tag{3.2.2}$$

where $C_{ij} = \sum_{m=0}^{15} h_{ij}(m)$ is a normalization constant. The multi-classifier $\mathcal{F}_{Q^2_{ij}}$ for a given value of the quantization step Q^2_{ij} is a "max-wins" multi-classifier with binary classifiers being C-SVMs with Gaussian kernel.

As was shown in the Section 3.1 on double-compression artifacts, the feature vector cannot distinguish the following cases: Q^1_{ij} is a divisor of Q^2_{ij}, $Q^1_{ij} = 1$, or $Q^1_{ij} = Q^2_{ij}$, because the normalized histograms are almost identical. Consequently, three cases are classified into one common class $Q^1_{ij} = Q^2_{ij}$. As will be seen in the section with experimental results, this is a fundamental limitation of the detector. Fortunately, it does not present problem for subsequent steganalysis, since the histograms of double-compressed DCT coefficients are almost the same.

3.2.3.2. *Matching the closest standard quantization matrix.* Once the primary quantization steps for modes $\mathcal{L}$ are detected, the closest standard quantization matrix has to be found. Denoting the estimated and the true primary quantization steps as $\hat{Q}^1_{ij}$ and Q^1_{ij}, respectively, the closest standard quantization matrix is obtained by the maximum likelihood principle as

$$\hat{Q} \quad = \quad \arg\max_{Q \in \mathcal{T}} \prod_{i,j \in \mathcal{L}} P(\hat{Q}^1_{ij} | Q_{ij}, Q^2_{ij}),$$

where $\mathcal{T}$ is the set of all standard quantization matrices. Since the number of quality factors is finite and the calculation of the likelihoods is fast, the maximum can be found by an exhaustive search over all $Q \in \mathcal{T}$. The conditional probabilities $P(\hat{Q}^1_{ij} | Q_{ij}, Q^2_{ij})$ are the probabilities that the classifier detects the primary quantization step $\hat{Q}^1_{ij}$ when the correct primary quantization step is Q_{ij} and the secondary quantization step is Q^2_{ij}. They are estimated empirically on images from the training set.

It is possible to incorporate a priori knowledge about the distribution of primary quantization tables into the estimation procedure and use a MAP estimator. This a priory information could be obtained by crawling the web and collecting the statistics about the JPEG quality tables. In this dissertation, however, this approach is not pursued.

3.3. Experimental results

In this section, experimental results of double-compression detector, primary quantization step detector, and primary quality factor estimator are presented and compared to selected prior art. All results in this section, including the prior art evaluation, were calculated on a database described in Section 2.5.

3.3.1. Double-compression detector. The training set of soft-margin SVMs (one for quality factor 75, and one for quality factor 80) consisted of 10000 examples of single-compressed images (cover images and images embedded by the F5, MBS1, MBS2, JP Hide&Seek, OutGuess, and Steghide steganographic algorithms, all with same probabilities) and 10000 examples of double-compressed images (cover images and images embedded by F5 and OutGuess). The hyper-parameters C and γ were determined by a grid-search on the multiplicative grid

$$(C, \gamma) \in \left\{(2^i, 2^j) | i \in \{0, \dots, 19\}, j \in \{-7, \dots, 5\}\right\},$$

combined with 5−fold cross-validation (the details about the search for optimal hyper-parameters can be found in Appendix A.4).

Figure 3.1 shows the accuracy of the DC detector on double-compressed JPEG images from the testing set. We can see that the accuracy on cover images and images embedded by OutGuess is very good. The accuracy on F5 images is worse, especially on images containing longer messages. This loss of accuracy is caused by the alteration of the shape of histograms of DCT coefficients by the shrinkage effect of F5[3]. As the primary quality factor increases, artifacts of double-compression are becoming more subtle and the accuracy of the detector decreases, which is to be expected.

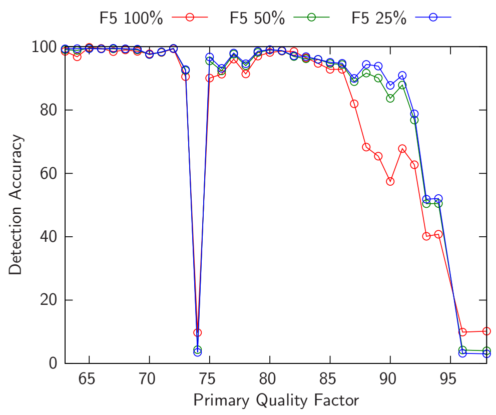
In Figure 3.1, we can observe sharp drops in the accuracy of the detector on images with primary quality factors 96 and 98 and on images with primary quality factor 74 and secondary quality factor 75. These sharp drops correspond to situations when the histograms of DCT coefficients are not affected by double-compression—all primary quantization steps for frequencies from $\mathcal{L}$ are divisors of the secondary quantization steps. The quantization steps for all 9 frequencies from $\mathcal{L}$ for standard matrices with quality factors 96 and 98 are all ones. Similarly, the quantization steps in the standard quantization matrices with quality factors 74 and 75 satisfy $Q_{ij}(74) = Q_{ij}(75), (i, j) \in \mathcal{L}$. Consequently, the decision of the detector is correct, since in these cases, the DCT coefficients in $\mathcal{L}$ are not double-compressed. Images with these combinations of quality factors were not used in the training set

Figure 3.2 shows the accuracy of the double-compression detector on single-compressed JPEG images embedded by various steganographic algorithms. Almost all of the tested steganographic algorithms preserve the histogram of DCT coefficients, which helps the detector to maintain its good accuracy. The only exception are images fully embedded by F5, which was already commented upon above. The detection accuracy on images embedded by F5 with shorter messages, is comparable to other algorithms. This improvement in accuracy is attributed to the decreased number of embedding changes due to matrix embedding.

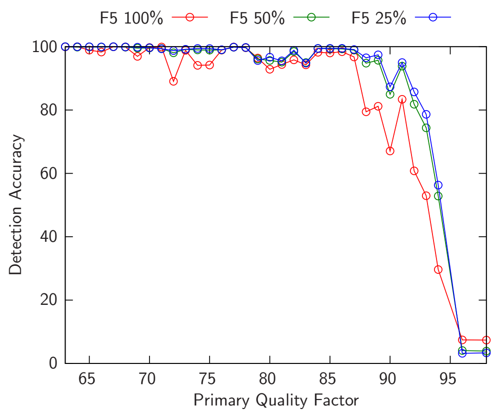
3.3.1.1. *Comparison with Benford features.* As was mentioned in Section 3.1.2, Shi et al. [**22**] proposed to use the histogram of the distribution of the first digit of DCT coefficients as a feature vector for a classifier detecting double-compression. In order to compare Benford features to Multiple-counting features described in Section 3.2.2, a $C-$SVM classifier was trained on each feature set. Since Benford features are not designed to deal with stego images, both classifiers were trained on cover images with the (secondary) quality factor 75. The size of the training set was $2 \times 3400 = 6800$ examples.

Table 3.2 shows the detection accuracy of both classifiers calculated on images from the

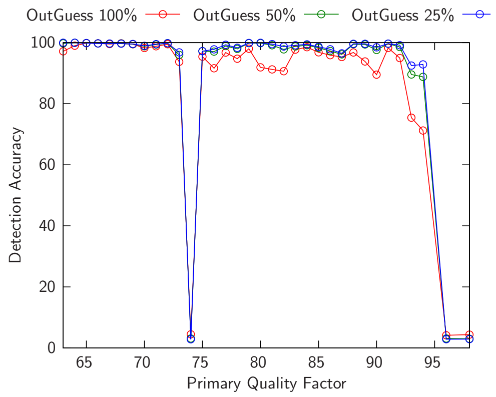
[3]The embedding operation of the F5 algorithm [**72**] always decreases the absolute value of DCT coefficients. Since only non-zero DCT coefficients are used to read the message, when a ± 1 is changed to zero, it is set to zero and next non-zero DCT coefficient is used instead. F5's embedding operation substantially increases the number of zeros in the histogram of DCT coefficients. This effect on the histogram is known as shrinkage.



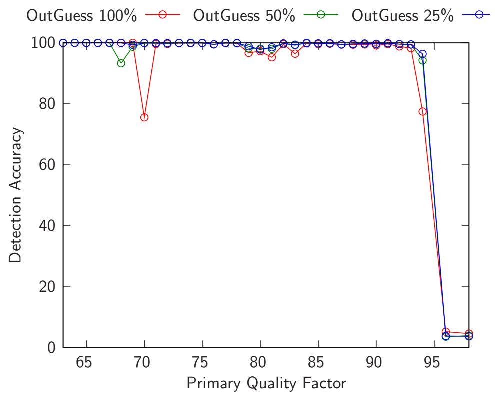
(a) F5, secondary quality factor 75



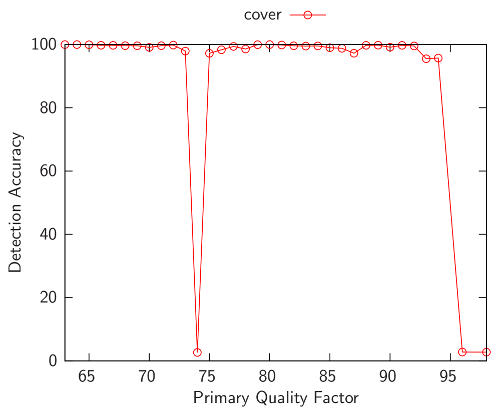
(b) F5, secondary quality factor 80



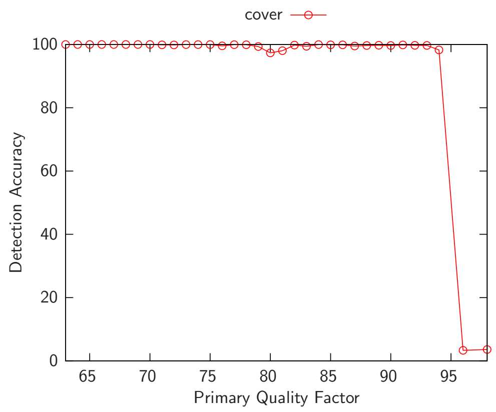
(c) OutGuess, secondary quality factor 75



(d) OutGuess, secondary quality factor 80



(e) Cover, secondary quality factor 75



(f) Cover, secondary quality factor 80

FIGURE 3.1. Accuracy of the double-compression detector for secondary quality factors 75 and 80 on double-compressed cover images and images embedded with F5 and OutGuess algorithms. Images with primary quality factor equal to the secondary quality factor are not double-compressed, which means that in this case, the correct answer of the detector is single-compressed. Graphs are drawn with respect to the primary quality factor.

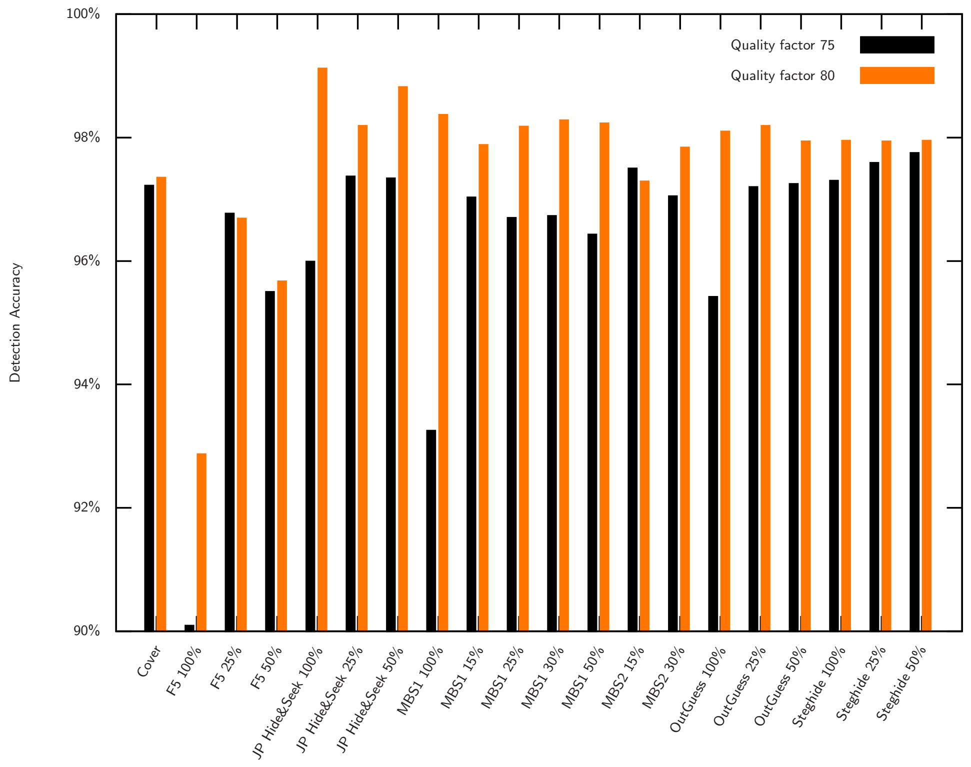


FIGURE 3.2. Accuracy of the double-compression detector on single-compressed JPEG images with quality factors 75 and 80. Note that the range of Y axis is $[90\%, 100\%]$.

	Feature set	
images	Benford	Multiple
Single-compressed	61.74%	98.64%
Double-compressed	30.91%	97.11%

TABLE 3.2. Accuracy of double-compression detectors employing Benford and Multiple-counting features. Since Benford features are not designed to deal with stego images, both detectors were trained and tested on *cover* images only.

testing set. Again, double-compressed images with primary quality factors 74, 96, and 98 were excluded, because DCT coefficients with spatial frequencies in $\mathcal{L}$ are not technically double-compressed (they exhibit divisor effect in all cases). Table 3.2 shows that while the performance of the Benford features on the used database of cover images is close to random guessing with bias towards the single-compressed class, the accuracy of Multiple-counting features is about 98%.

3.3.2. Estimation of primary quantization coefficients. As described in Section 3.2.3.1, the detector of primary quantization step is implemented by a collection of "max-wins" multi-classifiers, where each multi-classifier consists of the set of soft-margin Support Vector Machines ($C-$SVM) with the Gaussian kernel. The training set for each $C-$SVM contained 20000 examples—10000 from each class. The hyper-parameters C and

SQS	4		5		6		7		8	
PQS	SVM	NN	SVM	NN	SVM	NN	SVM	NN	SVM	NN
1	96.0%	**98.6%**	85.2%	**87.4%**	**92.4%**	91.4%	79.8%	**95.7%**	67.8%	**90.9%**
2	96.2%	**98.6%**	**95.3%**	74.7%	**92.1%**	91.8%	**86.3%**	74.1%	69.3%	**91.4%**
3	**98.8%**	96.9%	**98.7%**	87.2%	**93.6%**	90.1%	**88.2%**	77.8%	**77.8%**	52.8%
4	95.7%	**98.7%**	**96.8%**	94.4%	**98.6%**	90.3%	**90.7%**	77.0%	71.1%	**91.8%**
5	**99.8%**	95.0%	84.3%	**86.4%**	**95.3%**	91.0%	**96.2%**	81.4%	**95.1%**	65.2%
6	**99.1%**	98.4%	**99.4%**	85.7%	**91.6%**	91.0%	**89.9%**	89.9%	90.7%	**94.0%**
7	**99.5%**	98.9%	**99.4%**	90.3%	**98.5%**	96.4%	80.0%	**95.6%**	**83.6%**	59.0%
8	**99.8%**	99.8%	**98.8%**	97.0%	**99.5%**	96.0%	**95.4%**	88.2%	67.0%	**91.2%**
9	—	—	98.3%	**98.6%**	**97.2%**	95.8%	**98.6%**	84.6%	**92.2%**	81.3%
10	—	—	**99.7%**	—	**99.8%**	—	**98.7%**	—	**93.6%**	—
11	—	—	—	—	**92.0%**	—	—	—	**97.9%**	—
12	—	—	—	—	**97.3%**	—	—	—	**99.0%**	—

TABLE 3.3. Accuracy of Neural Network (NN) and Support Vector Machine (SVM) primary quantization steps detectors on *COVER* images from the testing set. PQS and SQS stand for primary and secondary quantization steps, respectively.

SQS	4		5		6		7		8	
PQS	SVM	NN	SVM	NN	SVM	NN	SVM	NN	SVM	NN
1	95.2%	**98.5%**	86.7%	**87.9%**	91.0%	**90.9%**	78.7%	**95.7%**	66.0%	**90.3%**
2	95.5%	**98.5%**	**84.1%**	45.1%	90.9%	**91.6%**	**65.3%**	44.5%	66.6%	**90.3%**
3	**95.1%**	67.4%	**94.1%**	59.2%	**92.4%**	89.7%	**81.9%**	52.1%	**72.0%**	36.7%
4	94.2%	**98.6%**	**95.1%**	71.8%	**94.6%**	59.4%	**83.4%**	52.1%	70.6%	**90.6%**
5	**99.4%**	78.4%	83.9%	**86.3%**	**94.0%**	70.6%	**91.3%**	51.5%	**87.6%**	40.4%
6	**99.3%**	76.8%	**98.2%**	66.4%	88.4%	**89.9%**	**85.0%**	70.8%	**83.6%**	68.2%
7	**99.5%**	61.7%	**99.4%**	61.7%	**97.2%**	81.0%	77.2%	**94.3%**	**79.2%**	42.0%
8	**99.4%**	72.1%	**98.9%**	70.2%	**99.4%**	59.2%	**93.7%**	58.3%	63.4%	**88.8%**
9	—	—	**97.5%**	72.3%	**97.7%**	80.8%	**97.3%**	58.9%	**87.5%**	58.2%
10	—	—	**99.2%**	—	**99.5%**	—	**98.7%**	—	**91.1%**	—
11	—	—	—	—	**90.4%**	—	—	—	**96.9%**	—
12	—	—	—	—	**96.0%**	—	—	—	**98.8%**	—

TABLE 3.4. Accuracy of Neural Network (NN) and Support Vector Machine (SVM) primary quantization steps detectors on *COVER* and *STEGO* images from the testing set. PQS and SQS stand for primary and secondary quantization steps, respectively.

γ were selected as a point from the multiplicative grid

$$(C,\gamma) \in \left\{(2^i, 2^j) | i \in \{4,\ldots,18\}, j \in \{-8,\ldots,6\}\right\}$$

with the smallest error estimated by 5-fold cross-validation.

Tables 3.3 and 3.4 compare the accuracy of the SVM-based primary quantization step detector with the Neural Network (NN) detector[4] from [**40**] on images from the testing set. The comparison is done for the secondary quantization steps 4, 5, 6, 7, and 8. The NN detector detects only the quantization steps in the range $[1, 9]$. Because the SVM detector

[4]The trained detector was kindly provided to us by the authors of [**40**].

was trained on cover and stego images, and the NN detector was trained on cover images only, the results on a mixed database of cover and stego images (Table 3.4) and on cover images only (Table 3.3) are showed. In most cases, the SVM based detector outperformed the NN detector. The rare occasions when the NN detector gave better results correspond to the situation when the primary quantization step was a divisor of the secondary step. As explained in Section 3.2.3.1, incorrect primary step detection in these cases has virtually no influence on steganalysis.

3.3.3. Estimation of the standard quantization matrix. The estimator of the standard quantization matrix requires the knowledge of the conditional probabilities $P(\hat{Q}^1_{ij}|Q^1_{ij}, Q^2_{ij})$ describing the accuracy of the detector of the primary quantization steps. These probabilities were evaluated empirically from images from the training set, as noted in Section 3.2.3.2.

Figure 3.3 shows the accuracy calculated on images from the testing set as a function of the true primary quality factor. As can be seen, the accuracy is not much affected by embedding. The detection on stego images embedded by F5 is worse (especially on fully embedded images) due to F5's shrinkage effect.

With the exception of images embedded by OutGuess with primary quality factor 75 and secondary quality factor 80, which is discussed later, all sharp drops in accuracy have the same cause. As explained in Section 3.2.3.1, if the primary quantization step Q^1_{ij} is a divisor of the secondary quantization step Q^2_{ij}, it is detected by default as Q^2_{ij}. Let us assume that Q and Q' are two primary quantization matrices for which

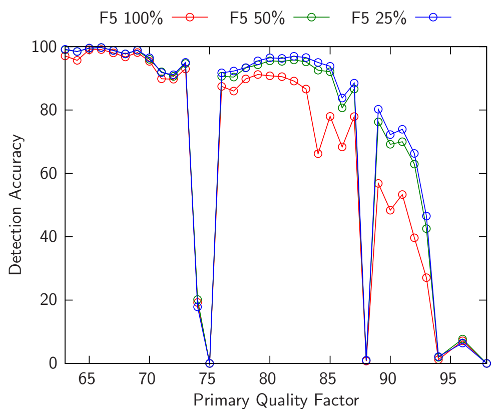
$$Q_{ij} \neq Q'_{ij} \Rightarrow Q_{ij}|Q^2_{ij} \text{ and } Q'_{ij}|Q^2_{ij}, \text{ for } (i,j) \in \mathcal{L}.$$

Let us further assume that for instance $\prod_{i,j\in\mathcal{L}} P(\hat{Q}^1_{ij}|Q_{ij}, Q^2_{ij}) > \prod_{i,j\in\mathcal{L}} P(\hat{Q}^1_{ij}|Q'_{ij}, Q^2_{ij})$. When detecting images with primary quantization matrix Q' (if all quantization steps are detected correctly), the ML estimator will incorrectly output Q instead of Q' because Q has larger likelihood. This failure is, fortunately, not going to impact subsequent steganalysis because when the primary quantization steps are divisors of the secondary quantization step, the impact of double-compression is negligible.

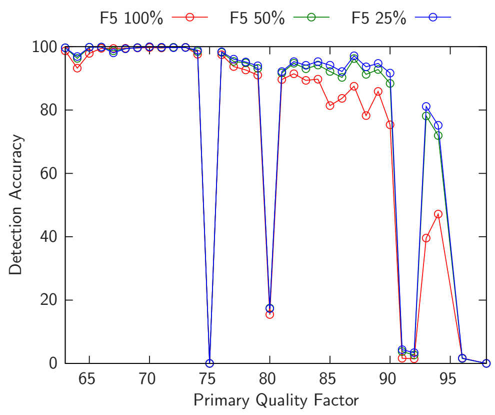
This phenomenon is illustrated on an example of images with the primary quality factor 88 and the secondary quality factor 75. Most of the time, the primary quality factor is estimated as 89. Let us denote the quantization matrices corresponding to quality factors 89, 88, and 75 as $Q(89)$, $Q(88)$, and $Q(75)$, respectively. By examining the quantization steps of $Q(89)$ and $Q(88)$ for frequencies $(i,j) \in \mathcal{L}$ (see figure 3.4) we observe that $Q(88)$ and $Q(89)$ only differ when $(i,j) = (0,1)$, in which case $Q^1_{01}(89) = 2$, $Q^1_{01}(88) = 3$, and $Q^2_{01}(75) = 6$. If all primary quantization steps are correctly detected ($\hat{Q}^1_{01}$ is detected as 6), then the estimator of the primary quality factor will prefer the quality factor 89 over 88 because the conditional probability $P(\hat{Q}^1_{01} = 6|Q^1_{01} = 2, Q^2_{01} = 6)$ is larger than $P(\hat{Q}^1_{01} = 6|Q^1_{01} = 3, Q^2_{01} = 6)$ and all other involved probabilities are the same.

The drop in accuracy on images embedded by OutGuess with the primary quality factor 85 and the secondary quality factor 75 is caused by the effect of embedding. The majority of incorrectly estimated images have the primary quality factor estimated as 84 instead of 85. The difference between the quantization matrices $Q(84)$ and $Q(85)$ is for frequency $(0,1)$, where $Q_{01}(84) = 4$ and $Q_{01}(85) = 3$. Because $Q_{01}(75) = 6$, this is not the case of divisors discussed above. From Figure 3.3(c), we see that the accuracy of estimation improves on images with shorter messages, which confirms the hypothesis about the influence of embedding.

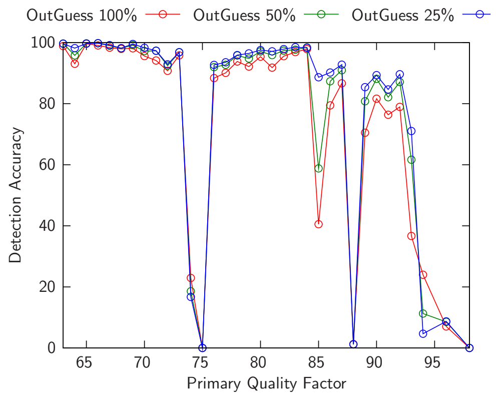
Table 3.5 shows the average decrease over all PQF in the detection accuracy when the PQF estimator was first applied only to cover images and then only to stego images. Because



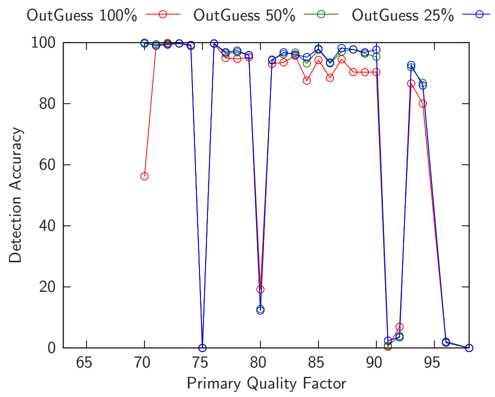
(a) F5, secondary quality factor 75



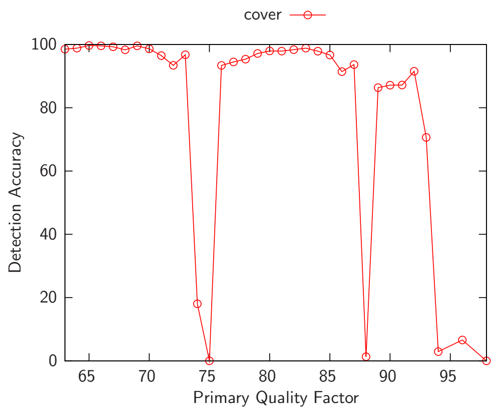
(b) F5, secondary quality factor 80



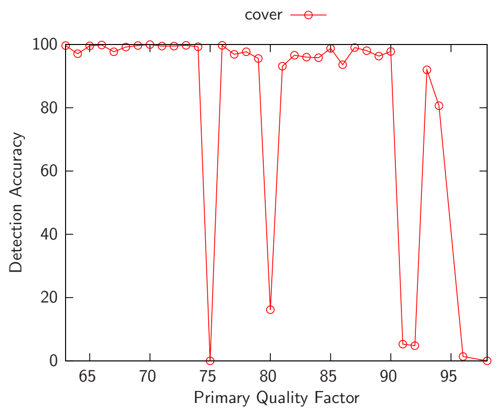
(c) OutGuess, secondary quality factor 75



(d) OutGuess, secondary quality factor 80



(e) Cover, secondary quality factor 75



(f) Cover, secondary quality factor 80

FIGURE 3.3. Accuracy of primary quality factor estimator for secondary quality factors 75 and 80 on double-compressed cover images and images embedded with F5 and OutGuess algorithms. Graphs are drawn with respect to the true primary quality factor. The graph showing OutGuess images with secondary quality factor 80 starts from the primary quality factor 70 because OutGuess fails to embed message into images with combination of primary quality factors $63, \ldots, 69$ and secondary quality factor 80.

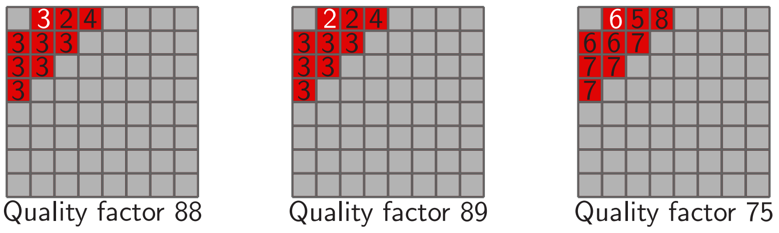


FIGURE 3.4. Quantization steps of standard quantization matrices from quality factors $75, 88$, and 89 on modes from the set $\mathcal{L}$

	SQF 75		SQF 80	
Algorithm	Mean	Std	Mean	Std
F5 100%	11.07	14.24	6.70	11.17
F5 50%	4.82	7.52	2.14	3.33
F5 25%	3.82	6.30	1.39	2.40
OutGuess 100%	5.12	12.08	3.76	8.55
OutGuess 50%	2.06	6.91	0.36	1.84
OutGuess 25%	0.37	1.67	0.11	1.48

TABLE 3.5. Mean and standard deviation of the drop in accuracy of the Primary Quality Factor estimator when applying it only to stego images and only to cover images.

F5 changes the histogram of DCT coefficients, the accuracy of the PQF estimator is worse for F5 embedded images than for OutGuess, which preserves the global histogram. The accuracy of the PQF estimator is expected to be even lower for steganographic techniques that significantly modify the histograms. It is unlikely, however, that such techniques will ever be developed because steganography significantly disturbing the first order statistics would likely be detectable using other means.

CHAPTER 4

Feature set for Blind Steganalysis

This chapter describes the Merged feature set used by the blind steganalyzer. The merged features set is created by extending the DCT feature set introduced by Fridrich [**17**] completed with calibrated Markov feature set (original Markov feature set was introduced by Shi et al. [**61**] is not calibrated). By comparing the Merged feature set to other feature sets it is shown, that the Merged feature set represents nowadays state of the art feature set for blind steganalysis of JPEG images.

The merged feature set is calculated from luminance part of JPEG image only. It's description starts with the calibration, which is an essential procedure used in the calculation of features. Than, feature sets of prior art together with their modifications are introduced, namely original DCT feature set together with its extended version, and Markov features with their calibrated version. Finally, Merged feature set is presented and compared to the prior art on selected problems of targeted and blind steganalysis.

4.1. Calibration

Calibration is a process used to estimate macroscopic properties of the cover image from the image under inspection J. Denoting the calibrated version of image as $\hat{J}$ and the feature extraction function as $\mathbf{f}$, the calibrated feature is obtained as the difference $\mathbf{f}(J) - \mathbf{f}(\hat{J})$. The calibration has a two-fold effect. First, it makes features $\mathbf{f}$ approximately zero mean on the set of cover images

$$E_{\mathsf{J}\sim P_c}[\mathbf{f}(\mathsf{J}) - \mathbf{f}(\hat{\mathsf{J}})] \approx 0.$$

Second, it decreases features variance

$$\mathrm{Var}_{\mathsf{J}\sim P_c}[\mathbf{f}(\mathsf{J}) - \mathbf{f}(\hat{\mathsf{J}})]$$

because it suppresses the sensitivity of feature extraction function $\mathbf{f}$ to image content (J denotes the random variable on the space of all images).

Figure 4.1 shows the calibration of a single-compressed JPEG image. The inspected JPEG image J is decompressed to the spatial domain, cropped by a few pixels (in our implementation the cropping is by $4, 4$ pixels) in both directions, and compressed again with the same quantization matrix as the original image J. The important part of the calibration process is cropping, which produces a new image visually similar to the original. This new

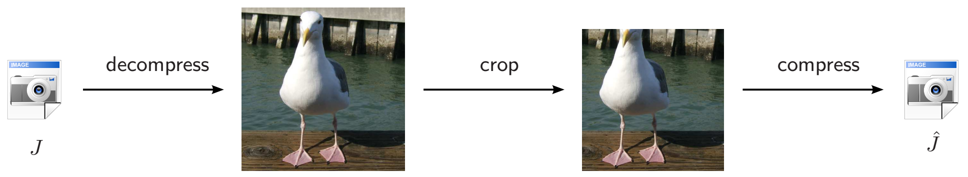


FIGURE 4.1. Calibration of the single-compressed stego image. Stego JPEG image J is decompressed to the spatial domain, cropped by a few pixels in both directions, and compressed again with the same quantization matrix as the original stego image J.

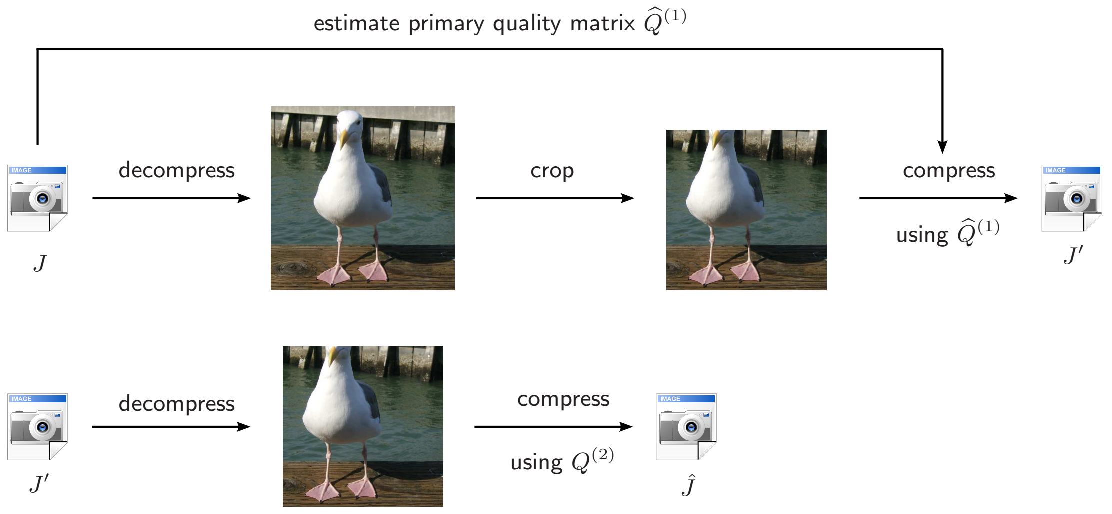

estimate primary quality matrix $\widehat{Q}^{(1)}$
decompress
crop
compress
using $\widehat{Q}^{(1)}$
J
J'
decompress
compress
using $Q^{(2)}$
J'
$\hat{J}$

FIGURE 4.2. Calibration of double-compressed image. The stego image J is decompressed to the spatial domain, cropped and compressed with the primary (cover) quantization matrix $\hat{Q}^{(1)}$. This new image $J^{'}$ is the estimate of the image before embedding. $J^{'}$ is again decompressed and compressed with the secondary quantization matrix $Q^{(2)}$, which yields into calibrated image $\hat{J}$.

image does not see the previous JPEG compression and possible steganography, because the 8×8 DCT grid is "out of sync" due to cropping. The subsequent compression with the same quantization matrix as was used to compress J produces the "calibrated" image $\hat{J}$ with most macroscopic features similar to the original cover image. Note that geometrical transformations other than cropping by 4 columns, can also be used, for example, a slight rotation, resizing, or random warping as performed in the attack on watermarking schemes called Stirmark [**39**].

If the stego image has been double-compressed during embedding (as it is the case for F5 and OutGuess), the calibration as described above would output an approximation to the single-compressed cover image instead of the approximation of double-compressed cover image. This discrepancy may lead to very inaccurate steganalysis results, which is described in [**17**] (an experimental verification is presented in Sections 5.3.2 and 6.2.2). The proper way to calibrate double-compressed JPEG images is to estimate the primary quantization matrix (for example by method presented in Chapter 3) and mimic the double compression. Calibration of double-compressed JPEG image is shown in Figure 4.2. The stego image J is decompressed to the spatial domain, cropped by a few pixels and compressed with the estimated primary (cover) quantization matrix $\hat{Q}^{(1)}$. This new image $J^{'}$ is the estimate of the image *before* embedding. $J^{'}$ is again decompressed to the spatial domain and compressed with the secondary quantization matrix $Q^{(2)}$, which yields the calibrated image $\hat{J}$.

4.2. Original DCT feature set and Extended DCT feature set

The *original DCT features* (originally published in [**17**]) are constructed using 23 feature extraction functionals $\mathbf{f}$ that produce a scalar, vector, or a matrix when applied to the stego image. The dimensionality of the original DCT features was aggressively decreased by calculating individual the calibrated features as the difference $\mathbf{f}(J) - \mathbf{f}(\hat{J})$, if $\mathbf{f}$ is a scalar, or as an L_1 norm $\|\mathbf{f}(J) - \mathbf{f}(\hat{J})\|_{L_1}$ if $\mathbf{f}$ is a vector or a matrix. The functionals are defined as follows.

Features are calculated only from the *luminance* part of the JPEG image. Let the luminance of a stego JPEG file be represented by a DCT coefficient array $D_{ij}(k)$, $i, j = 0, \dots, 7$, $k = 1, \dots, n_B$, where $D_{ij}(k)$ denotes the (i, j)-th quantized DCT coefficient in the k-th block (there are total of n_B blocks).

The first set of functionals captures first order statistic of DCT coefficients $D_{ij}(k)$. Under the assumption that $D_{ij}(k)$ are realizations of an iid random variable, their complete statistical description can be captured by their probability density function.

The first functional is the global histogram $\mathbf{H}$ of all $64 \times n_B$ luminance DCT coefficients

$$\mathbf{H} = (H_L, \dots, H_R), \tag{4.2.1}$$

where $L = \min_{i,j,k} D_{ij}(k)$, $R = \max_{i,j,k} D_{ij}(k)$.

Due to the different quantization steps and frequencies on different DCT modes, it is expected that the probability density function of DCT coefficients on different modes will be different as well. To capture these differences, the next 5 functionals are histograms

$$\mathbf{h}^{ij} = (h^{ij}_L, \dots, h^{ij}_R), \tag{4.2.2}$$

of coefficients on 5 individual DCT modes $(i, j) \in \{(0, 1), (1, 0), (2, 0), (1, 1), (0, 2)\}$. Histograms on higher modes are not used, because due to high quantization steps, $D_{ij}(k)$ are frequently zeros which makes their statistics insufficient for steganography.

The next 11 functionals are dual histograms represented by 8×8 matrices $\mathbf{g}^d_{ij}$, $i, j = 0, \dots, 7$, $d = -5, \dots, 5$

$$\mathbf{g}^d_{ij} = \sum_{k=1}^{n_B} \delta(d, D_{ij}(k)), \tag{4.2.3}$$

where $\delta(x, y) = 1$ if $x = y$ and 0 otherwise. The dual histogram captures the distribution of a given coefficient value r among different DCT modes. Note that if a steganographic method preserves all individual histograms, it also preserves all dual histograms and vice versa.

The assumption of DCT coefficients being independent made above does not hold for real world images. Coefficients of real images exhibit dependencies between neighboring blocks. The last 6 functionals capture this inter-block dependency among the coefficients. The first functional is the variation V

$$V = \frac{\sum\limits_{i,j=0}^{7} \sum\limits_{k=1}^{|\mathbf{I}_r|-1} |D_{ij}(\mathbf{I}_r(k)) - D_{ij}(\mathbf{I}_r(k+1))| + \sum\limits_{i,j=0}^{7} \sum\limits_{k=1}^{|\mathbf{I}_c|-1} |D_{ij}(\mathbf{I}_c(k)) - D_{ij}(\mathbf{I}_c(k+1))|}{|\mathbf{I}_r| + |\mathbf{I}_c|}, \tag{4.2.4}$$

where $\mathbf{I}_r$ and $\mathbf{I}_c$ denote the vectors of block indices $1, \dots, n_B$ while scanning the image by rows and by columns, respectively. The embedding changes increase the variation V, as they add entropy to the array of DCT coefficients $D_{ij}(k)$.

The next three functionals are calculated from the co-occurrence matrix of neighboring DCT coefficients

$$\begin{aligned} N_{00} =& \mathbf{C}_{0,0}(J) - \mathbf{C}_{0,0}(\hat{J}) \\ N_{01} =& \mathbf{C}_{0,1}(J) - \mathbf{C}_{0,1}(\hat{J}) + \mathbf{C}_{1,0}(J) - \mathbf{C}_{1,0}(\hat{J}) + \mathbf{C}_{-1,0}(J) - \mathbf{C}_{-1,0}(\hat{J}) + \mathbf{C}_{0,-1}(J) - \mathbf{C}_{0,-1}(\hat{J}) \\ N_{11} =& \mathbf{C}_{1,1}(J) - \mathbf{C}_{1,1}(\hat{J}) + \mathbf{C}_{1,-1}(J) - \mathbf{C}_{1,-1}(\hat{J}) + \\ & + \mathbf{C}_{-1,1}(J) - \mathbf{C}_{-1,1}(\hat{J}) + \mathbf{C}_{-1,-1}(J) - \mathbf{C}_{-1,-1}(\hat{J}), \end{aligned} \tag{4.2.5}$$

where

$$\mathbf{C}_{st} = \frac{1}{|\mathbf{I}_r| + |\mathbf{I}_c|} \left[\sum_{i,j=0}^{7} \sum_{k=1}^{|\mathbf{I}_r|-1} \delta\left(s, D_{ij}(\mathbf{I}_r(k))\right) \delta\left(t, D_{ij}(\mathbf{I}_r(k+1))\right) + \sum_{i,j=0}^{7} \sum_{k=1}^{|\mathbf{I}_c|-1} \delta\left(s, D_{ij}(\mathbf{I}_c(k))\right) \delta\left(t, D_{ij}(\mathbf{I}_c(k+1))\right) \right]. \tag{4.2.6}$$

The co-occurrences capture transition probabilities of corresponding DCT coefficient between neighboring blocks in horizontal and vertical direction.

The last two functionals called blockiness are scalars calculated from the decompressed JPEG image. They represent an integral measure of discontinuity along the borders of 8×8 blocks

$$B_\alpha = \frac{\sum_{i=1}^{\lfloor (M-1)/8 \rfloor} \sum_{j=1}^{N} |\mathbf{c}_{8i,j} - \mathbf{c}_{8i+1,j}|^\alpha + \sum_{j=1}^{\lfloor (N-1)/8 \rfloor} \sum_{i=1}^{M} |\mathbf{c}_{i,8j} - \mathbf{c}_{i,8j+1}|^\alpha}{N \lfloor (M-1)/8 \rfloor + M \lfloor (N-1)/8 \rfloor}. \tag{4.2.7}$$

In (4.2.7), M and N are image height and width in pixels and $\mathbf{c}_{ij}$ are greyscale values of the decompressed JPEG image, $\alpha = 1, 2$.

The original motivation for using the L_1 norm to form the calibrated DCT features is the reduction of their dimensionality. It is apparent, however, that by using the L_1 norm, some information potentially useful for steganalysis is lost. By replacing the L_1 norm with a higher-dimensional alternative, more information should be preserved giving better classification results at the expense of increased dimensionality. Replacing the L_1 norm directly with the difference, however, is not feasible because the feature set dimensionality would substantially increase and there would be too many features holding little information (e.g., histogram bins for large values of DCT coefficients). This would eventually negatively affect the performance and increase the complexity of the classifier. In order to alleviate the information loss due to using the L_1 norm and to keep the dimensionality of features "reasonable," *Extended DCT features* uses differences instead of the L_1 norm.

For the global histogram functional $\mathbf{H}$ and for 5 histograms of individual DCT modes $\mathbf{h}^{ij}$, $(i,j) \in \{(0,1),(1,0),(2,0),(1,1),(0,2)\}$, the differences of elements in the range $[-5,+5]$ are used. Thus, the histogram features are

$$\mathbf{H}_l(J_1) - \mathbf{H}_l(J_2),\ l \in \{-5,\dots,5\},$$

$$\mathbf{h}_l^{ij}(J_1) - \mathbf{h}_l^{ij}(J_2),\ l \in \{-5,\dots,5\}.$$

For the dual histogram functionals $\mathbf{g}^d, d \in \{-5,\dots,+5\}$, the difference of the 9 lowest AC modes

$$\mathbf{g}_{ij}^d(J_1) - \mathbf{g}_{ij}^d(J_2),\ (i,j) \in \{(1,0),(2,0),(3,0),(0,1),(1,1),(2,1),(0,2),(1,2),(0,3)\}$$

are used. For the co-occurrence matrix functionals, the central elements in the range $[-2,+2] \times [-2,+2]$ are used. This yields 25 features

$$\mathbf{C}_{st}(J_1) - \mathbf{C}_{st}(J_2),\ (s,t) \in [-2,+2] \times [-2,+2].$$

The rationale behind restricting the range of the differences between functionals to a small interval around zero is that the DCT coefficients follow a generalized Gaussian distribution centered around zero. Thus, the central part of the functionals holds the most useful information for steganalysis.

Functional	Dimensionality
Global histogram $\mathbf{H}_l$	11
5 AC histograms $\mathbf{h}_l^{ij}$	5×11
11 Dual histograms $\mathbf{g}_{ij}^{d}$	11×9
Variation V	1
2 Blockiness B_α	2
Co-occurrence matrix $\mathbf{C}_{st}$	25

TABLE 4.1. Extended DCT feature set with 193 features.

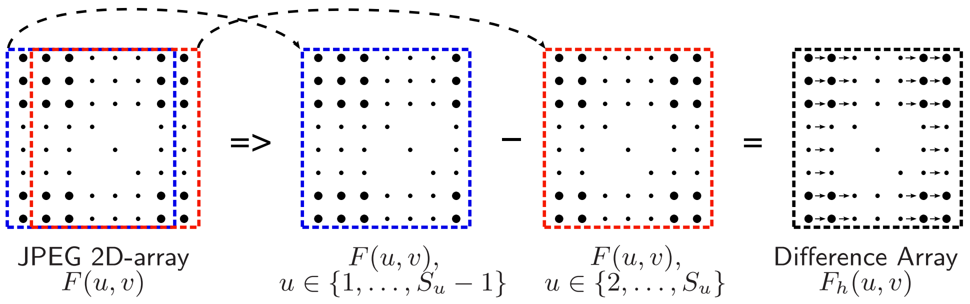


FIGURE 4.1. Schematic of the formation of difference array $F_h(u,v)$ along horizontal axis. S_u denotes width of the JPEG image in pixels. Arrows depicted in the difference array matrix $F_h(u,v)$ show the direction of the transition probabilities in the matrix $\mathbf{M}_h$.

After the L_1 norm is replaced by the differences, the dimensionality of the feature set increases from 23 to 193 (see Table 4.1).

4.3. Original and Calibrated Markov features

The DCT transformation applied to an 8×8 pixel block is orthogonal. Transformed and quantized DCT coefficients $D_{ij}(k)$ exhibit weak dependency between neighboring coefficients. This dependency is exploited by the *Markov feature set* proposed in [**61**], which models the differences between absolute values of neighboring DCT coefficients as a Markov process. The feature calculation starts by forming the matrix $F(u,v)$ of absolute values of DCT coefficients $D_{ij}(k)$ in the image (in the original publication [**61**], $F(u,v)$ is called JPEG 2D-array). The DCT coefficients in $F(u,v)$ are arranged in the same way as pixels in the image by replacing each 8×8 block of pixels with the corresponding block of DCT coefficients[1]. Next, four difference arrays are calculated along four directions: horizontal, vertical, diagonal, and minor diagonal (further denoted as $F_h(u,v)$, $F_v(u,v)$, $F_d(u,v)$, and $F_m(u,v)$ respectively)

$$\begin{aligned}
F_h(u,v) &= F(u,v) - F(u+1,v),\\
F_v(u,v) &= F(u,v) - F(u,v+1),\\
F_d(u,v) &= F(u,v) - F(u+1,v+1),\\
F_m(u,v) &= F(u+1,v) - F(u,v+1).
\end{aligned}$$

An example of formation of difference array in horizontal direction $F_h(u,v)$ is shown on Figure 4.1. From these difference arrays, four transition probability matrices $\mathbf{M}_h, \mathbf{M}_v, \mathbf{M}_d, \mathbf{M}_m$

[1]Assuming k is indexing DCT blocks along a horizontal scan and n_h is the number of 8×8-blocks in one horizontal line, then $F(u,v) = |D_{ij}(k)|$, where $i = u \bmod 8$, j=v mod 8, and $k = n_b \cdot \left\lfloor \frac{v}{8} \right\rfloor + \left\lfloor \frac{u}{8} \right\rfloor + 1$. $\lfloor\cdot\rfloor$ denotes floor operation.

are constructed as

$$\begin{aligned}
\mathbf{M}_h(i,j) &= \frac{\sum_{u=1}^{S_u-2}\sum_{v=1}^{S_v}\delta(F_h(u,v)=i, F_h(u+1,v)=j)}{\sum_{u=1}^{S_u-1}\sum_{v=1}^{S_v}\delta(F_h(u,v)=i)},\\
\mathbf{M}_v(i,j) &= \frac{\sum_{u=1}^{S_u}\sum_{v=1}^{S_v-2}\delta(F_v(u,v)=i, F_v(u,v+1)=j)}{\sum_{u=1}^{S_u}\sum_{v=1}^{S_v-1}\delta(F_v(u,v)=i)},\\
\mathbf{M}_d(i,j) &= \frac{\sum_{u=1}^{S_u-2}\sum_{v=1}^{S_v-2}\delta(F_d(u,v)=i, F_d(u+1,v+1)=j)}{\sum_{u=1}^{S_u-1}\sum_{v=1}^{S_v-1}\delta(F_d(u,v)=i)},\\
\mathbf{M}_m(i,j) &= \frac{\sum_{u=1}^{S_u-2}\sum_{v=1}^{S_v-2}\delta(F_m(u+1,v)=i, F_m(u,v+1)=j)}{\sum_{u=1}^{S_u-1}\sum_{v=1}^{S_v-1}\delta(F_m(u,v)=i)},
\end{aligned}$$

where S_u and S_v denote the dimensions of the image and $\delta = 1$ if and only if its argument(s) are satisfied. The transition probability matrices $\mathbf{M}_h, \mathbf{M}_v, \mathbf{M}_d$, and $\mathbf{M}_m$ are calculated along the same direction as the difference arrays $F_h(u,v)$, $F_v(u,v)$, $F_d(u,v)$, and $F_m(u,v)$. The range of differences between absolute values of neighboring DCT coefficients could be quite large. If the matrices $\mathbf{M}_h, \mathbf{M}_v, \mathbf{M}_d, \mathbf{M}_m$ were taken directly as features, the dimensionality of the feature set would be prohibitively high. Thus, the authors proposed to only use the central $[-4, +4]$ portion of the matrices with the caveat that the values in the difference arrays $F_h(u,v)$, $F_v(u,v)$, $F_d(u,v)$, and $F_m(u,v)$ larger than 4 were set to 4 and values smaller than -4 were set to -4 prior to calculating $\mathbf{M}_h, \mathbf{M}_v, \mathbf{M}_d, \mathbf{M}_m$. Thus, all four matrices have the same dimensions 9×9 and the number of features is $4 \times 81 = 324$.

The Markov features as proposed in [**61**] are uncalibrated (here they are called *original Markov Features*). Because the calibration is known to improve features' sensitivity to embedding while reducing image-to-image variations, the calibration was incorporated into the process of calculating the features. As expected, calibration significantly improves the steganalytic performance of Markov features. Let $\mathbf{M}$ denote the transition probability matrix in a specific direction. The *calibrated Markov features* are formed by differences $\mathbf{M}^{(c)} = \mathbf{M}(J) - \mathbf{M}(\hat{J})$, where J is the stego image and $\hat{J}$ its calibrated version. The dimension of the calibrated Markov feature set, $\mathbf{M}_h^{(c)}, \mathbf{M}_v^{(c)}, \mathbf{M}_d^{(c)}, \mathbf{M}_m^{(c)}$, remains the same as its original version.

4.4. Merged feature set

While the Markov features capture residual *intra-block* dependencies among DCT coefficients of neighboring spatial frequencies within the same 8×8 block, the extended DCT features capture *inter-block* dependencies between DCT coefficients. Since the dependencies between DCT coefficients captured by Markov and extended DCT features are complementary, it makes sense to merge both feature sets. Another incentive for merging them is the observation (see Sections 4.5.1 and 4.5.2) that both feature sets complement each other in performance. For example, the extended DCT features are better in detecting JP Hide&Seek, while the calibrated Markov features are better in detecting F5.

A direct combination of both feature sets would produce a 517-dimensional feature vector. To reduce the dimensionality, the average $\overline{\mathbf{M}} = (\mathbf{M}_h^{(c)} + \mathbf{M}_v^{(c)} + \mathbf{M}_d^{(c)} + \mathbf{M}_m^{(c)})/4$ of all four calibrated matrices forming Markov features is used instead. This feature vector $\overline{\mathbf{M}}$ has dimensionality 81. Experiments showed that averaged features $\overline{\mathbf{M}}$ accompanied with extended DCT features produced very similar performance to the full version $\mathbf{M}_h^{(c)}, \mathbf{M}_v^{(c)}, \mathbf{M}_d^{(c)}, \mathbf{M}_m^{(c)}$. After merging the 193 extended DCT features with the 81 averaged calibrated Markov features, the dimension of the resulting Merged feature set became $193 + 81 = 274$.

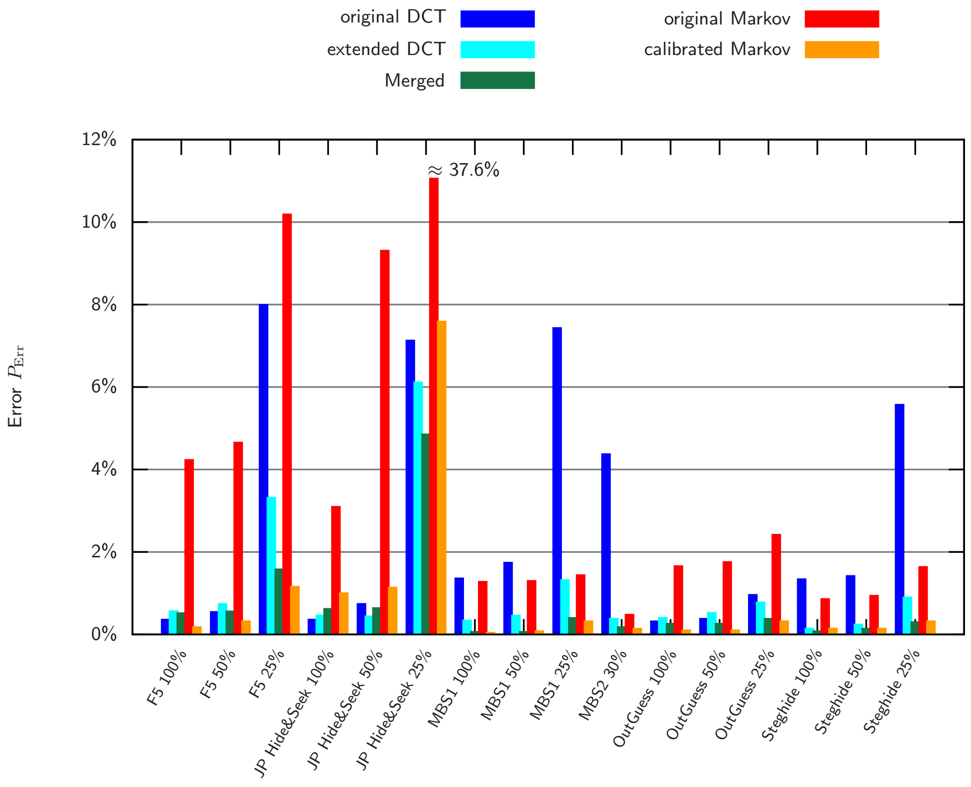


FIGURE 4.1. Comparison of DCT, extended DCT, original Markov, calibrated Markov, and Merged feature sets on the problem of detection of particular steganographic algorithm (binary classification). The error is measured as $P_{err} = 0.5(P_{FP} + P_{MD})$ on images from the testing set. All images are single-compressed JPEG images with quality factor 75.

4.5. Comparison of features

This section presents the comparison of all feature sets described above, namely the original DCT features, extended DCT features, original Markov features (without calibration), Markov features with calibration and Merged features, on two selected problems. The first problem is the targeted attack on six steganographic algorithms (F5, JP Hide&Seek, Model Based Steganography without deblocking (MBS1), Model Based Steganography with deblocking (MBS2), OutGuess, and Steghide). The second problem is a blind steganalysis problem, when Eve wants to detect the algorithm used for embedding. In the latter problem, the image is classified into one of many classes (one class for each steganographic algorithm + cover class). The same set of algorithms as in the case of targeted attacks was used. It is obvious that the second problem is substantially more difficult than the first one, which pronounces the differences between feature sets.

4.5.1. Binary classifiers. The targeted classifiers were trained on 3400 examples of cover images and 3400 examples of images embedded by 6 steganographic algorithms with 3 different message lengths (details about training and testing database can be found in Section 2.5). All images were single-compressed JPEG images with quality factor 75, which is the default quality factor of the implementation of OutGuess. Total of $5\times6 = 30$ classifiers were created — one classifier for each combination of the feature set and the steganographic algorithm.

Classifiers were implemented by soft-margin Support Vector Machines (C-SVM) with Gaussian kernel. All feature sets were individually scaled to interval $[-1, +1]$ with parameters derived on the training set. The hyper-parameters of the C-SVMs were determined as the point with the smallest error estimated by 5-fold cross-validation on the unbounded multiplicative grid

$$(C, \gamma) \in \{(2^i, 2^j) | i \in \mathcal{Z}, j \in \mathcal{Z}\}. \tag{4.5.1}$$

The details of the methodology are described in Appendix A.4.

The testing images had the same processing history (quality factor, steganographic algorithm, length of messages) as images in the training set, but they were created from a disjoint set of 2506 raw images.

Figure 4.1 shows the error rate of the binary classifiers measured as $P_{\text{Err}} = \frac{1}{2}(P_{\text{FP}} + P_{\text{MD}})$ of all 30 binary classifiers. If we compare the error of the original DCT features to the error of the extended DCT features on images containing short messages embedded by advanced steganographic algorithms (F5, MBS2, Steghide), we can see that the L_1 norm used in the original DCT features discards a lot of useful information. In some cases, the error of extended DCT features is almost $7\times$ lower than the error of the original version.

As is mentioned above, DCT and Markov features capture different dependencies within JPEG image. The effect is demonstrated by complementary performance of both feature sets, as the extended DCT feature set is better in detecting JP Hide&Seek and OutGuess, while the Markov feature set is better in detecting Steghide.

Another interesting fact is the effect of calibration on Markov features. The calibrated Markov features offer significantly better performance than the uncalibrated version. This improvement is most visible on images embedded with JP Hide&Seek with message length of 25% of image capacity. The error of the classifier employing the original Markov features is 47%, while the error of the classifier employing calibrated Markov features is 7%.

Figure 4.1 also shows superior performance of Merged features, which is to be expected as the Merged feature set benefits from Extended DCT and calibrated Markov feature sets.

4.5.2. Multi-classifier. The task of assigning stego images to corresponding stego class is more difficult than the targeted steganalysis presented in the previous section. Consequently, it should be expected that the differences between individual feature sets will be more pronounced. Five multi-classifiers, each employing one feature set, were constructed. All multi-classifiers were trained to classify images into one out of 7 classes: cover, F5, OutGuess, JP Hide&Seek, MBS1, MBS2, and Steghide.

Multi-classifiers are implemented as "max-wins" multi-classifiers described in Section A.5. Training and testing of individual binary classifiers (each multi-classifier consists of $\binom{7}{2} = 21$ binary classifiers) was done according to the same method as in Section 4.5.1.

Figure 4.2 shows the detection accuracy of the multi-classifiers. The detection accuracy is defined as the rate of assigning an image to the correct class (for example cover image to cover class, image embedded by F5 algorithm to F5 class, etc.). The results of the comparison of multi-classifiers are similar to results of targeted steganalysis in the previous section, but the differences between feature sets are now more pronounced. The complementary performance of extended DCT and calibrated Markov feature sets is more visible, as the former performs better on JP Hide&Seek, MBS2, OutGuess and Steghide, while the latter performs better on F5 and MBS1 images.

As expected, the Merged feature set offers the best performance despite the fact that its dimensionality is lower than the dimensionality of the original and calibrated Markov feature sets.

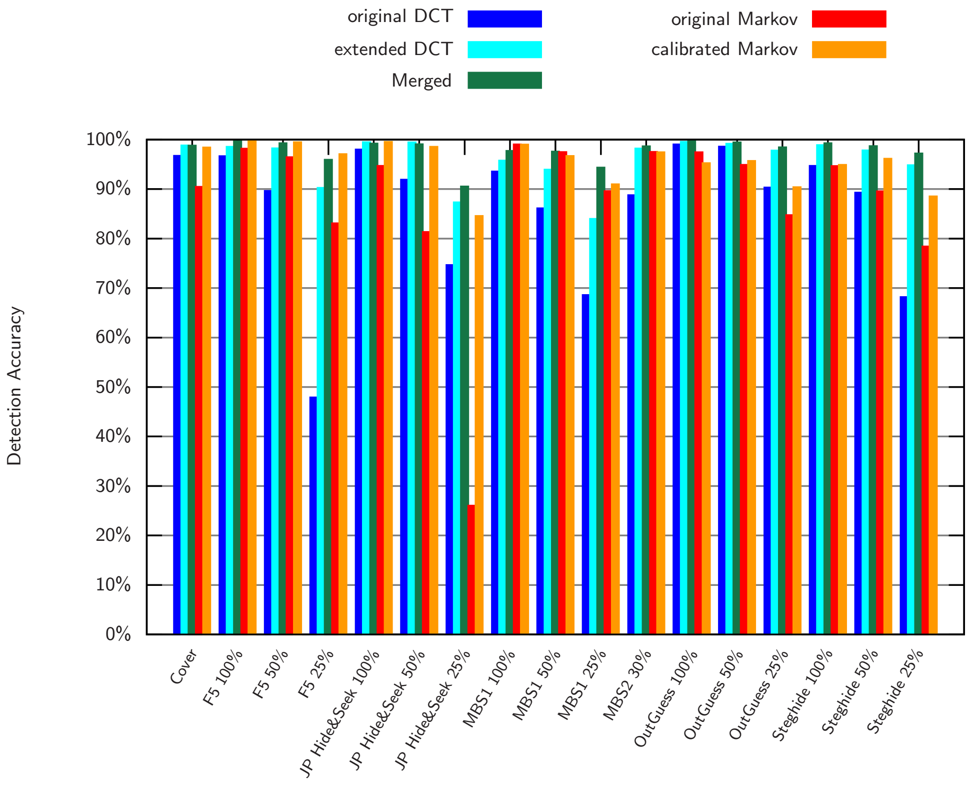


Figure 4.2. Comparison of DCT, extended DCT, original Markov, calibrated Markov, and Merged feature sets on multi-classification problem of detection of steganographic algorithm. Graphs shows detection accuracy, which is the rate of correctly assigning an image to the class according to the algorithm used for embedding, calculated on images from the testing set. All images are single-compressed JPEG images with quality factor 75.

CHAPTER 5

Blind Steganalyzer for JPEG Images

This chapter finishes the description of the blind steganalyzer (see Section 2.3 for the outline of the blind steganalyzer) by first describing the classifier for single-compressed and double-compressed images. After both classifiers are introduced, the performance of the whole blind steganalyzer is shown. The performance is evaluated on several testing sets in order to evaluate the accuracy of the blind steganalyzer under different conditions.

Images in the first testing set have the same processing history as images in the training set. They were created from 2506 raw images never used in any form in the training set. The first testing set contains $\approx 2,606,240$ images.

Images in the second testing set were created from 300 JPEG images taken by the 6Mpix Fuji E550 camera. Images were embedded with the same combination of steganographic algorithms and message lengths as images in the previous testing set. The second testing set was used to estimate the performance of the blind steganalyzer on images with non-standard quantization matrices.

The third testing set contains images embedded by algorithms not used during training, namely JSteg, MMx [**37**], and –F5 [**21**]. The purpose of the third testing set is to estimate accuracy on stego images produced by unknown algorithm.

The modules for correct handling of double-compressed images in the blind steganalyzer substantially increase its complexity. Section 5.4 presents simple experiments showing that the increased complexity is rewarded by increased accuracy.

5.1. Classifier for Single-Compressed JPEG images

The classifier for single-compressed JPEG images detects cover images and images embedded by the following 6 steganographic algorithms: F5 [**72**], OutGuess [**55**], JP Hide&Seek [1], MBS1 [**56**], MBS2 [**57**], and Steghide [**27**]. It consists of a bank of SVM-based multi-classifiers $\mathcal{S}_q$ trained for each value of the quality factor

$$q \in \mathcal{Q}_{34} = \{63, 64, \dots, 93, 94, 96, 98\}.$$

To avoid confusion of multi-classifier $\mathcal{S}_q$ targeted to a given quality factor q with the classifier referring to the bank of multi-classifiers $\{\mathcal{S}_q | q \in \mathcal{Q}_{34}\}$, targeted multi-classifier is always denoted by $\mathcal{S}_q$. All multi-classifiers $\mathcal{S}_q$ use the Merged feature set (Chapter 4) employing *single-compressed calibration*.

The modular approach utilizing bank of multi-classifiers $\mathcal{S}_q$ offers several advantages over monolithic design with one big multi-classifier for all quality factors $\mathcal{Q}_{34}$. First, the training of the bank of multi-classifiers is by order of 34^2 faster. This is because the complexity of SVM training is approximately $O(N^3)$ with N being the number of training examples. Second, it is possible to extend the classifier to other quality factors $q \notin \mathcal{Q}_{34}$ without the need to change already trained multi-classifiers $\mathcal{S}_q, q \in \mathcal{Q}_{34}$. Although it is possible to use a multi-classifier $\mathcal{S}_q$ to classify images with other quality factors $q' \neq q$, [**50**] shows that the accuracy of classification decreases.

[1]Can be obtained from: `http://linux01.gwdg.de/~alatham/stego.html`

The multi-classifiers $\mathcal{S}_q$ are implemented as "max-wins" multi-classifiers (Appendix A.5) consisting of $\binom{n_{cl}}{2}$ binary C-SVMs for every pair of classes, where n_{cl} is the number of classes into which we wish to classify (in this case $n_{cl} = 7$ resulting in 21 binary classifiers). All C-SVMs (Appendix A) use the Gaussian kernel $\exp(-\gamma\|x - y\|^2)$. The hyper-parameters (C, γ) of the C-SVMs were determined as

$$\arg \min_{(C,\gamma)\in\mathcal{G}} \hat{E}(C, \gamma),$$

where $\hat{E}(C, \gamma)$ denotes the error estimated by 5−fold cross-validation for hyper-parameters (C, γ), and $\mathcal{G}$ is a unbounded multiplicative grid

$$\mathcal{G} = \{(2^i, 2^j)|i \in \mathcal{Z}, j \in \mathcal{Z}\}.$$

The grid-search on unbounded grid is described in detail in Appendix A.4.

In blind steganalysis, one might be interested in lowering the false positive rate — the probability that a cover image is detected as stego. This can be done either by shifting the threshold of SVMs, or better with weighted SVMs described in Appendix A.3. Since weighted SVMs with Gaussian kernel have 3 hyper-parameters $(C_{\text{pos}}, C_{\text{neg}}, \gamma)$, the grid-search for a suitable triplet $(C_{\text{pos}}, C_{\text{neg}}, \gamma)$ becomes computationally very expensive. From this reason, this approach was not employed in the design of the blind steganalyzer.

The features of all classifiers were preprocessed so that all elements of the feature vectors were in the interval $[-1, +1]$. The scaling coefficients were always derived from the training set.

5.1.1. Training sets of multi-classifiers $\mathcal{S}_q$. The max-wins multi-classifier $\mathcal{S}_q$ targeted to JPEG images with quality factor $q \in \mathcal{Q}_{34}$ employs $\binom{n_{class}}{2} = 21$ binary classifiers for every pair out of $n_{class} = 7$ classes (cover, F5, JP Hide&Seek, MBS1, MBS2, OutGuess and Steghide). Before the training sets used to train single-compression multi-classifiers $\mathcal{S}_q$ can be described, it is important to emphasize the difference between multi-classifiers $\mathcal{S}_{75}, \mathcal{S}_{80}$, and other multi-classifiers $\mathcal{S}_q, q \in Q_{34}\backslash\{75, 80\}$. This difference reflects itself in the construction of the training sets. A careful examination of the path of a double-compressed image through the blind steganalyzer (Figure 2.1) shows that double-compressed images (they are assumed to have quality factor 75 or 80, see Section 2.1) incorrectly detected by the DC detector as single-compressed are presented to the multi-classifiers $\mathcal{S}_{75}, \mathcal{S}_{80}$ for single-compressed images. Since the DC detector is tuned to have a low false positive rate, which comes at the expense of a higher false negative rate (double-compressed images detected as single-compressed), these cases will not be scarce in real use.

For quality factors $q \in \mathcal{Q}_{34}\backslash\{75, 80\}$, the size of the training set for individual binary SVMs in multi-classifiers $\mathcal{S}_q$ is constrained by having only 3500 examples of cover and 3500 examples of MBS2 images (details of the database of images used for the experiment are in Section 2.5). In order to have at least elementary diversity and overcome issues with the number of examples being slightly smaller due to the failure of the embedding algorithm, the training set of each binary classifier consists of 3400 examples from each class (total number of examples in the training set was $2 \times 3400 = 6800$). For classes where there were examples with 3 message lengths corresponding to 100%, 50%, and 25% of the algorithm embedding capacity, examples were chosen randomly so that the training set contained an approximately equal number examples of each message length. For example, the training set for the binary classifier detecting cover and F5 images consisted of 3400 examples of covers, 1133 examples of F5 images with 100% message, 1133 examples of F5 images with 50% message, and 1134 examples of F5 images with 25% message.

As was mentioned above, the multi-classifiers $\mathcal{S}_{75}$ and $\mathcal{S}_{80}$ might encounter double-compressed images incorrectly classified by the DC detector as single-compressed. Thus, it seems natural that the training set for the multi-classifiers $\mathcal{S}_{75}$ and $\mathcal{S}_{80}$ should contain such images. These misclassified images increase the number of cover, F5, and OutGuess examples available for training. This makes possible to have a larger training set for all C-SVMs except for the C-SVMs detecting MBS2, where the limitation of having only 3500 examples remains. In practice, the training sets of all binary classifiers except the binary classifiers detecting MBS2 consisted of approximately 3×3400 examples from each class ($3 \times 3400 + 3 \times 3400 = 20400$ examples in total). The training set for the binary classifiers detecting the MBS2 algorithm consisted of 3400 examples from each class (2×6800 examples in total).

5.2. Classifier for Double-Compressed JPEG images

The classifier for double-compressed JPEG images classifies into three classes: cover images, and images embedded by F5 and OutGuess algorithms, because under the design assumptions stated in Section 2.1, only these two algorithms can produce double-compressed images. The construction of the multi-classifier for double-compressed images is very similar to the construction of the multi-classifier for single-compressed images described in Section 5.1. It consists of two max-wins SVM-based multi-classifiers, $\mathcal{D}_{75}, \mathcal{D}_{80}$, designed for secondary quality factors 75 and 80. The feature vector of the multi-classifiers is formed by Merged features with *double-compression calibration* (Chapter 4) augmented by the primary quality factor estimated by the Primary Quality Factor estimator (Section 3.2.3.2) as an additional feature. The dimension of the feature vector is 275, which is by 1 feature more than the dimension of the feature vector of the multi-classifier for single-compressed images.

The development of the multi-classifier for double-compressed JPEG images requires considerable computing power, because examples of all combinations of primary and secondary quality factors to be detected need to be prepared. This is why only two multi-classifiers were created. The primary quality factors were all from $\mathcal{Q}_{34}$.

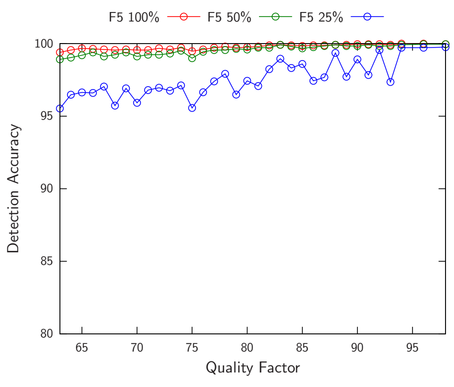
The multi-classifiers $\mathcal{D}_{75}, \mathcal{D}_{80}$ were created following the same procedures as the multi-classifiers $\mathcal{S}_q$.

5.2.1. Training sets of multi-classifiers $\mathcal{D}_{75}, \mathcal{D}_{80}$. Both "max-wins" multi-classifiers $\mathcal{D}_{75}$ and $\mathcal{D}_{80}$ employ only 3 binary classifiers. All images available for training were pre-classified by the double-compression detector in order to train the multi-classifiers $\mathcal{D}_{75}$ and $\mathcal{D}_{80}$ on double-compressed images *detected as* double-compressed. The training set for all 3 C-SVMs consisted of 10000 examples from each class (20000 examples total). The distribution of primary quality factors of images in the training sets followed the distribution determined by the double-compression detector. The training set did not contain any examples of single-compressed images detected as double-compressed.

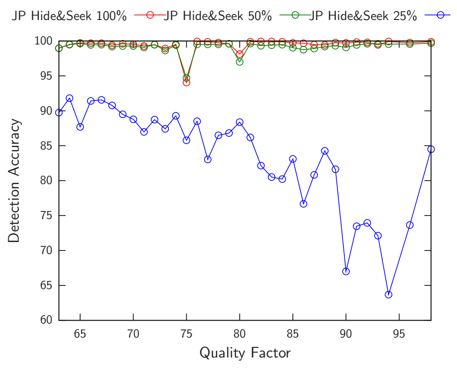
5.3. Experimental results from the Blind Steganalyzer

The accuracy of the blind steganalyzer was estimated on a testing set containing $|\mathcal{Q}_{34}| \times 17 \times 2506 + (|\mathcal{Q}_{34}| - 1) \times 2 \times 7 \times 2506 \approx 2,606,240$ images never seen by the classifier.

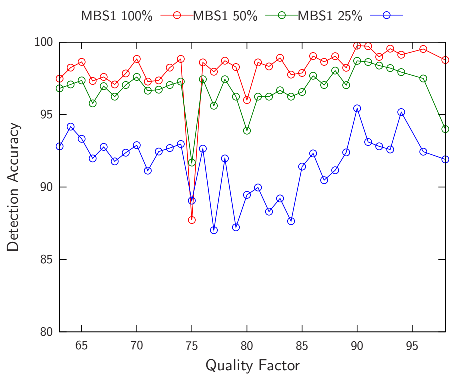
Figure 5.1 shows the detection accuracy (image is correctly assigned to the embedding algorithm) of the blind steganalyzer on single-compressed JPEG images plotted with respect to the quality factor. The accuracy on images containing longer messages (50% and more) is most of the time better than 97% and remains practically the same through the entire range of quality factors. As the embedded message becomes shorter, the accuracy decreases, which is to be expected. The accuracy on images embedded with MBS1, MBS2, JP Hide&Seek, and Steghide exhibits drops of the order of a few percents for quality factors 75 and 80 caused by incorrect detection of single-compressed images as double-compressed, (recall



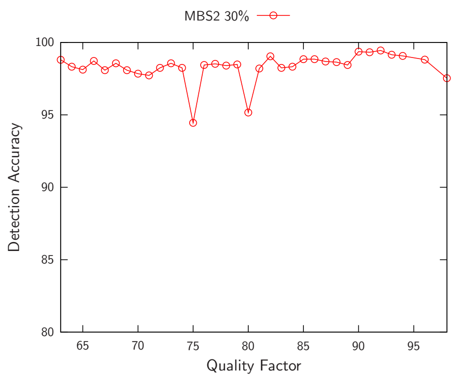
(a) F5



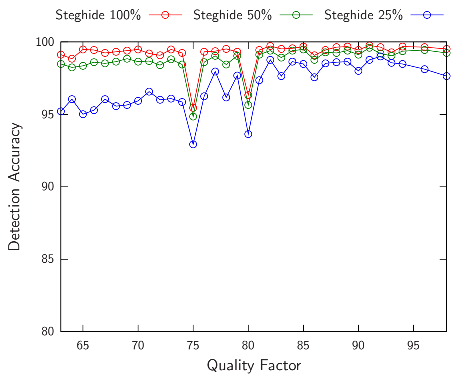
(b) JP Hide&Seek



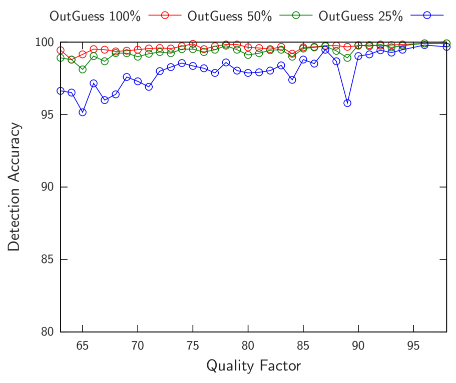
(c) MBS1



(d) MBS2



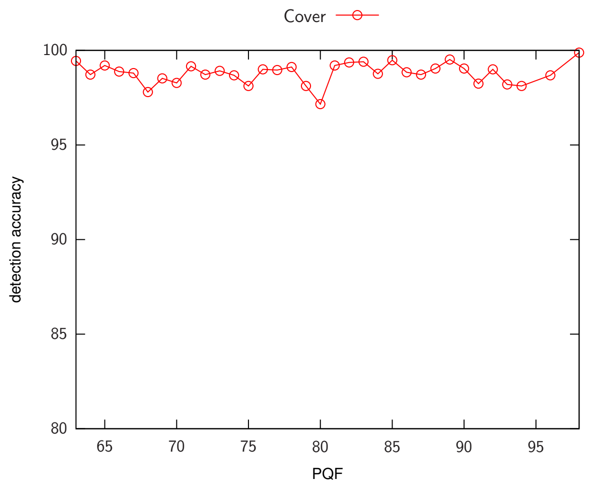
(e) Steghide



(f) OutGuess

FIGURE 5.1. Detection accuracy of the blind steganalyzer on single-compressed JPEG images. Detection accuracy is plot with respect to quality factor of JPEG images. Note the different y-axis scale for JP Hide&Seek.

that the double-compression detector is used only for secondary quality factors 75 and 80). Figure 3.2 confirms that, as MBS1 has one of the highest ratio of single-compressed images detected as double-compressed, and, indeed, the drop in the performance in Figure 5.1 is



(a) Cover

FIGURE 5.2. Detection accuracy of the multi-classifier on single-compressed cover images from the testing set.

the highest. This phenomenon underlines the importance of a low false positive rate of the double-compression detector, as already discussed in Chapter 3.

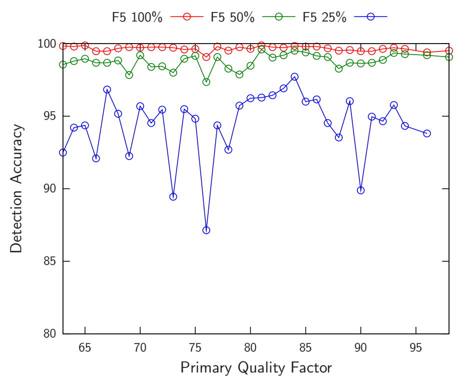
The overall false positive rate of the blind steganalyzer on single-compressed cover images is 1.2%, which is acceptable, considering the fact that the training did not include any mechanism to lower the false positive rate. The detection accuracy on single-compressed cover images plot against quality factor is shown in Figure 5.2. We can see that the detection accuracy does not change substantially with the quality factor.

Comparing the detection accuracy on double-compressed images shown in Figures 5.3 and 5.4 to the detection accuracy on single-compressed images (Figures 5.1, 5.2), it can be seen that the accuracy is not affected by double-compression to a greater extent. Moreover, it changes only little with the primary quality factor, which confirms the preposition made in Section 3.2.3.1 that the failure to correctly estimate the primary quality factor has negligible effect on steganalysis. The only exception are images embedded with a 25% message by the F5 algorithm. In this case, the performance is worse for double-compressed images. This loss in accuracy is due to the combined distortion caused by double compression and the small number of embedding changes due to matrix embedding.

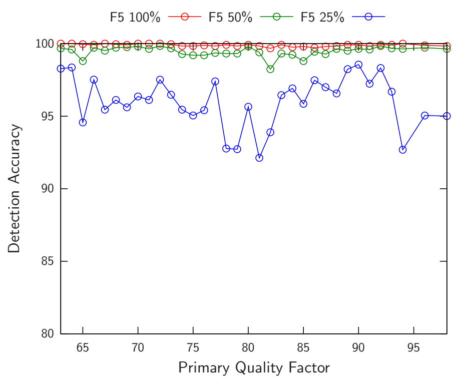
Tables 5.1,5.2 show confusion tables calculated on single and double-compressed images with secondary quality factors 75 and 80. The classification accuracy decreases as the embedded message gets shorter. At this point, it is important to point out certain fundamental limitations that cannot be overcome. In particular, it is not possible to distinguish between two algorithms that employ the same embedding mechanism by inspecting the statistics of DCT coefficients. For example, two algorithms that use LSB embedding along a pseudo-random path will be indistinguishable in the feature space. This phenomenon might be responsible for “merging” of the MBS1, MBS2, and Steghide classes.

5.3.1. Steganalysis of images with custom quantization matrix. The purpose of the experiment presented in this section is to evaluate the accuracy of the steganalyzer on images with custom (non-standard) quantization matrices. Such matrices are used in some consumer digital cameras.

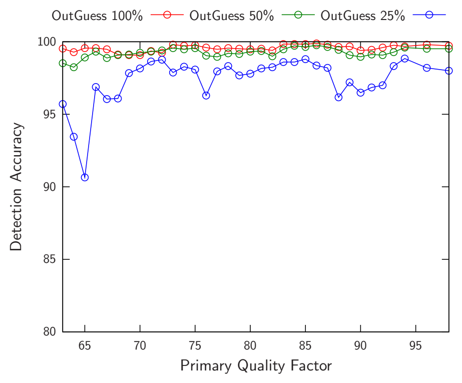
For this experiment, 300 JPEG images were collected from a 6-megapixel camera Fuji E550 with JPEG compression setting HQ. This camera is well suited for such a test, because



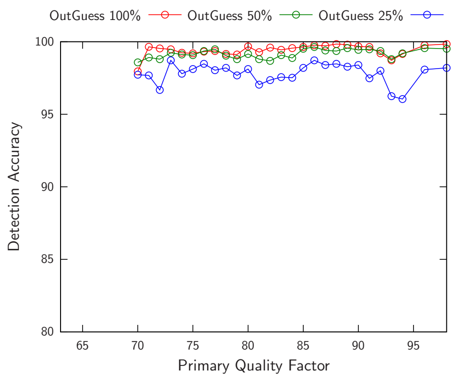
(a) F5, secondary quality factor 75



(b) F5, secondary quality factor 80



(c) OutGuess, secondary quality factor 75

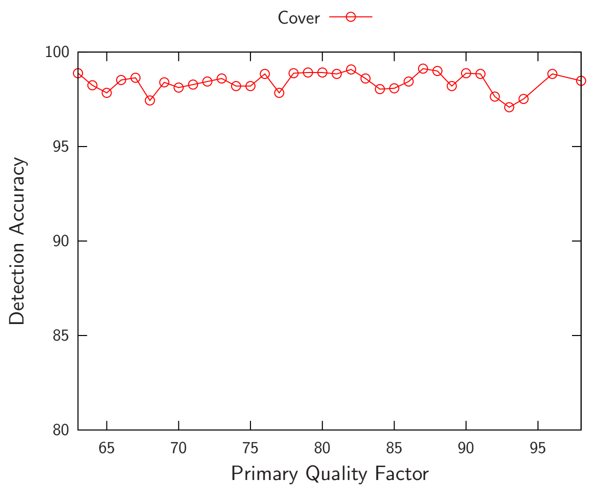


(d) OutGuess, secondary quality factor 80

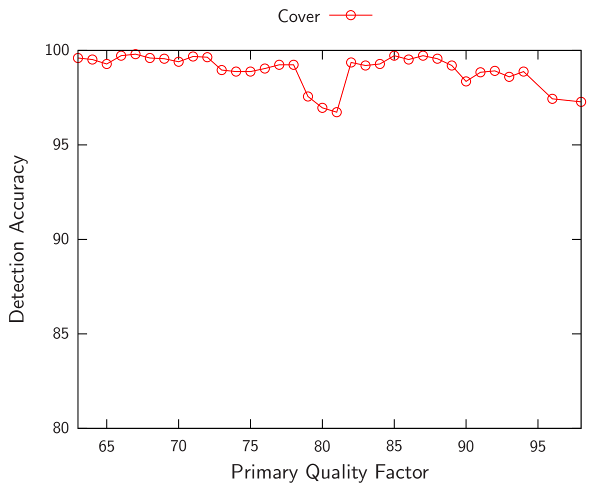
FIGURE 5.3. Accuracy of the multi-classifier on double-compressed JPEG images with secondary quality factors 75 and 80. The graph showing OutGuess images with secondary quality factor 80 starts from the primary quality factor 70 because OutGuess fails to embed message into images with combination of primary quality factors $63, \dots, 69$ and secondary quality factor 80.

it uses a wide range of custom quantization tables depending on the scene. Among the 300 images, there were total of 165 different quantization matrices. The average quality factor across all 300 images, determined as described in Section 3.2.1, was approximately 90. As before, 3 relative payloads, 25%, 50%, and 100% of the image capacity were embedded, using F5, JP Hide&Seek, MBS1, Steghide, and OutGuess, and 30% relative message length using MBS2. The stego images produced by JP Hide&Seek, Steghide, MBS1, and MBS2 retained their original custom quantization matrices, while images from F5 and OutGuess produced double-compressed images with their default quality factors.

The classification accuracy of the blind steganalyzer is presented in Table 5.3. The detection accuracy for stego images is quite good, but the detection suffers from an increased false positive rate. It is to be expected that the steganalyzer may not classify images with very high quality factors correctly. This limitation occurs due to the fact that at higher JPEG qualities more non-zero AC coefficients start appearing in medium and high frequencies. And if the match between the custom quantization matrix and the closest



(a) Cover, secondary quality factor 75



(b) Cover, secondary quality factor 80

FIGURE 5.4. Accuracy of the multi-classifier on double-compressed cover JPEG images with secondary quality factors 75 and 80.

standard matrix is poor in those bands, the features extracted by the steganalyzer will start statistically deviating from the training set, which will result in less reliable detection.

5.3.2. Is the special treatment of double-compressed images necessary? The mechanism of the blind steganalyzer for correct handling of double-compressed images consists of three tools: the double-compression detector, the primary quality factor estimator, and the multi-classifier for double-compressed images. A small experiment presented in this section shows that this increased complexity is indeed justified by the substantial increase of the accuracy of the classifier.

Total of 10000 images were randomly selected from the database of double-compressed testing images with the same fraction of cover images and images embedded by F5 and OutGuess algorithms with all three relative payloads. The secondary quality factor of all images was 75. Selected images were classified by the multi-classifier $\mathcal{S}_{75}$ targeted to single-compressed JPEG images with quality factor 75 (details about training can be found in Section 5.1). The resulting confusion matrix is shown in Table 5.4 and should be contrasted to Table 5.1, which demonstrates the classification accuracy that can be achieved by the

	Classified as						
	Cover	F5	JP H&S	MBS1	MBS2	OutGuess	Steghide
F5 100%	0.24%	**99.65%**	0.00%	0.00%	0.00%	0.10%	0.00%
JP H&S 100%	2.88%	2.52%	**94.05%**	0.00%	0.00%	0.56%	0.00%
MBS1 100%	0.16%	0.40%	0.00%	**87.72%**	1.44%	10.04%	0.24%
OutGuess 100%	0.04%	0.35%	0.00%	0.02%	0.01%	**99.57%**	0.01%
Steghide 100%	0.04%	0.32%	0.00%	0.08%	0.08%	4.03%	**95.45%**
F5 50%	0.81%	**98.79%**	0.00%	0.01%	0.01%	0.36%	0.02%
JP H&S 50%	2.48%	2.28%	**94.69%**	0.00%	0.00%	0.56%	0.00%
MBS1 50%	0.36%	0.20%	0.00%	**91.69%**	1.56%	5.64%	0.56%
OutGuess 50%	0.25%	0.42%	0.00%	0.02%	0.02%	**99.26%**	0.03%
Steghide 50%	0.44%	0.36%	0.00%	0.28%	0.04%	4.03%	**94.85%**
MBS2 30%	0.76%	0.24%	0.00%	0.76%	**94.45%**	3.67%	0.12%
F5 25%	4.60%	**94.47%**	0.04%	0.01%	0.01%	0.78%	0.10%
JP H&S 25%	10.98%	2.68%	**85.78%**	0.00%	0.00%	0.56%	0.00%
MBS1 25%	3.47%	0.60%	0.00%	**89.06%**	1.16%	2.56%	3.15%
OutGuess 25%	1.81%	0.61%	0.00%	0.02%	0.03%	**97.40%**	0.13%
Steghide 25%	3.04%	0.96%	0.00%	0.56%	0.20%	2.32%	**92.93%**
Cover	**98.40%**	0.95%	0.05%	0.01%	0.00%	0.59%	0.01%

TABLE 5.1. Confusion matrix for the multi-classifier trained for quality factor 75 tested on single and double-compressed 75-quality JPEG images. The first column contains the embedding algorithm and the relative message length. The remaining columns show the classification results. JP H&S is an abbreviation of JP Hide&Seek.

blind steganalyzer, which correctly handles double-compressed images. By comparing both tables it is clear that the classifier only trained on single-compressed images does not handle double-compressed images well. The reason for the poor performance on double-compressed images is that the calibration did not properly estimate the properties of the cover image. The striking difference in the accuracy justifies the increased complexity of the blind steganalyzer needed to correctly handle double-compressed images.

5.3.3. Novelty detection. Detecting steganographic algorithms unseen during training is an important property. To assess the ability of the blind steganalyzer to generalize to previously unseen stego methods, the steganalyzer was presented with stego images created by methods the steganalyzer was not trained on, namely Jsteg[2], the recently proposed MMx [**37**], and experimental –F5.

Jsteg uses simple LSB embedding in quantized DCT coefficients (coefficients 0 and 1 are skipped) along a pseudo-random path generated from a secret key. The MMx method is a more sophisticated algorithm that requires side information in the form of the uncompressed image. The MMx algorithm minimizes the combined distortion due to quantization and embedding in a combination with modified matrix embedding using Hamming codes.

–F5 employs entirely different embedding mechanism that does not resemble any mechanism used by the 6 stego-algorithms used during training. It embeds message bits into quantized DCT coefficients by changing their parity (LSB) along a pseudo-random path. If a parity of a DCT coefficient needs to be changed, instead of decreasing the absolute value of the coefficient as in F5, it is *increased*. This has a nice side effect of eliminating shrinkage from F5 (situation when a non-zero DCT coefficient is changed to zero), which

[2]Freely available for download at `http://zooid.org/~paul/crypto/jsteg/`.

	Classified as						
	Cover	F5	JP H&S	MBS1	MBS2	OutGuess	Steghide
F5 100%	0.07%	**99.89%**	0.00%	0.00%	0.00%	0.03%	0.00%
JP H&S 100%	0.84%	1.04%	**98.12%**	0.00%	0.00%	0.00%	0.00%
MBS1 100%	0.08%	0.44%	0.00%	**96.00%**	1.12%	1.88%	0.48%
OutGuess 100%	0.04%	0.46%	0.00%	0.01%	0.00%	**99.47%**	0.01%
Steghide 100%	0.04%	0.64%	0.00%	0.04%	0.04%	2.92%	**96.33%**
F5 50%	0.41%	**99.47%**	0.00%	0.00%	0.01%	0.10%	0.00%
JP H&S 50%	1.40%	1.56%	**97.00%**	0.00%	0.00%	0.04%	0.00%
MBS1 50%	0.36%	0.72%	0.00%	**93.89%**	1.96%	2.04%	1.04%
OutGuess 50%	0.23%	0.60%	0.00%	0.02%	0.02%	**99.11%**	0.02%
Steghide 50%	0.28%	0.80%	0.00%	0.08%	0.12%	3.08%	**95.65%**
MBS2 30%	1.36%	0.28%	0.00%	0.84%	**95.16%**	2.12%	0.24%
F5 25%	3.49%	**96.07%**	0.03%	0.00%	0.00%	0.39%	0.02%
JP H&S 25%	9.19%	2.32%	**88.38%**	0.00%	0.00%	0.12%	0.00%
MBS1 25%	2.08%	0.60%	0.00%	**89.46%**	1.48%	1.92%	4.47%
OutGuess 25%	1.33%	0.82%	0.02%	0.01%	0.03%	**97.75%**	0.04%
Steghide 25%	2.44%	1.74%	0.00%	0.22%	0.09%	1.87%	**93.64%**
Cover	**98.93%**	0.72%	0.18%	0.00%	0.00%	0.18%	0.00%

TABLE 5.2. Confusion matrix for the multi-classifier trained for quality factor 80 tested on single and double-compressed 80-quality JPEG images. The first column contains the embedding algorithm and the relative message length. The remaining columns show the classification results. JP H&S is an abbreviation of JP Hide&Seek.

	Cover	F5	JP H&S	MBS1	MBS2	OutGuess	Steghide
F5 100%	0.67%	99.33%	0.00%	0.00%	0.00%	0.00%	0.00%
JP H&S 100%	1.00%	0.33%	98.67%	0.00%	0.00%	0.00%	0.00%
MBS1 100%	0.00%	0.00%	0.00%	97.31%	2.69%	0.00%	0.00%
OutGuess 100%	0.00%	0.00%	0.00%	0.00%	1.67%	98.33%	0.00%
Steghide 100%	0.00%	0.00%	0.00%	1.67%	1.33%	0.00%	97.00%
F5 50%	2.00%	98.00%	0.00%	0.00%	0.00%	0.00%	0.00%
JP H&S 50%	0.33%	0.67%	99.00%	0.00%	0.00%	0.00%	0.00%
MBS1 50%	0.00%	0.00%	0.00%	98.65%	1.01%	0.00%	0.34%
OutGuess 50%	0.67%	0.00%	0.00%	1.67%	4.67%	92.67%	0.33%
Steghide 50%	0.00%	1.33%	0.00%	2.00%	0.33%	0.00%	96.33%
MBS2 30%	0.00%	0.00%	0.00%	15.49%	83.84%	0.00%	0.67%
F5 25%	7.00%	93.00%	0.00%	0.00%	0.00%	0.00%	0.00%
JP H&S 25%	5.33%	1.00%	93.67%	0.00%	0.00%	0.00%	0.00%
MBS1 25%	0.67%	0.67%	0.00%	90.91%	0.67%	0.00%	7.07%
OutGuess 25%	17.00%	6.67%	0.33%	0.67%	2.67%	62.67%	10.00%
Steghide 25%	3.00%	3.33%	0.00%	5.33%	0.00%	0.00%	88.33%
Cover 0%	90.67%	3.33%	6.00%	0.00%	0.00%	0.00%	0.00%

TABLE 5.3. Confusion matrix for stego images with non-standard quantization tables from Fuji E550. JP H&S is an abbreviation of JP Hide&Seek.

target	Cover	F5	JP H&S	MBS1	MBS2	OutGuess	Steghide
F5	13.32%	**83.45%**	0.24%	0.02%	0.07%	2.82%	0.08%
OutGuess	11.73%	3.69%	0.05%	0.19%	0.31%	**83.82%**	0.21%
Cover	**81.55%**	11.96%	0.37%	0.02%	0.09%	5.96%	0.04%

TABLE 5.4. Confusion matrix of single-compression multi-classifier $\mathcal{S}_{75}$ (Section 5.1) calculated on double-compressed images from the testing set. If we compare accuracy in this Table with accuracy of the blind steganalyzer correctly handling double-compressed images (Table 5.1), we can see that increased accuracy of the blind steganalyzer justifies its increased complexity. JP H&S is an abbreviation of JP Hide&Seek.

complicates the embedding mechanism of F5 and decreases its embedding efficiency (number of bits embedded per embedding change). Similar to F5, –F5 uses Hamming codes for matrix embedding to decrease the number of embedding changes.

–F5 is a poor steganographic algorithm. It can be shown that it introduces the largest combined distortion due to embedding and quantization [**21**] out of any embedding operation that changes a fraction of $\delta \geq 0$ coefficients towards zero and $1 - \delta$ away from zero (–F5 is obtained for $\delta = 0$ and F5 corresponds to $\delta = 1$). The comparison of the security of several steganographic algorithms presented in Section 7.1 shows that –F5 is indeed very detectable.

The JSteg and –F5 stego images were prepared from 2506 raw images used exclusively for testing, by embedding messages of length 100%, 50%, and 25% of their embedding capacity. The stego images for MMx were embedded with random messages of relative length 0.66, 0.42, and 0.26 bpac (bits per non-zero DCT coefficient). These payloads match the capacities determined by the co-dimension of the Hamming codes $[2^p - 1, 2^p - p - 1]$ for $p = 2, 3, 4$ used for matrix embedding. MM2, and MM3 stand for the version of the algorithm that allows up to two or three modifications per embedding block. The security improves with the number of allowed changes. The "MM1" algorithm would be very similar to F5, but it will not suffer from shrinkage effect, because coefficients with absolute values 1 would be possibly changed to coefficients with absolute value 2. The quality factor for all images was set to 75. All stego images were single-compressed.

Table 5.5 shows that Jsteg is very reliably detected using the blind steganalyzer even though Jsteg embedded images were not used for the classifier construction. It is interesting to note that Jsteg was mostly detected as F5 and OutGuess. Images embedded with the MMx algorithm were also reliably detected as stego and were assigned mostly to Model Based Steganography and Steghide. The missed detection rate for MMx quickly increases with decreasing message length due to the improved matrix coding scheme. Here, we remark that MMx can be detected more reliably using a targeted detector constructed from the same feature set (see the results in Section 7.1).

Quite surprisingly, –F5 is almost indictable at high embedding rates, while at low embedding rates it becomes detectable. In other words, the multi-classifier $\mathcal{S}_{75}$ completely failed to recognize images fully embedded with –F5 as containing stego content and instead classified them as covers. This is likely because the 6 individual binary classifiers distinguishing between covers and a stego method were all adjusted for low false positive rate, which is a necessity for any practical steganalytic tool. Thus, because stego images fully embedded with –F5 did not resemble any of the stego images on which the classifier was trained, most of those 6 binary classifiers (cover vs. stego) conservatively assigned the image to the cover class. Even though the decisions of the remaining binary classifiers were biased towards F5 and OutGuess, stego classes usually did not get enough votes. In cases when

	Cover	F5	JP H&S	MBS1	MBS2	OutGuess	Steghide
Jsteg 100%	0.20%	57.91%	0.00%	0.00%	0.04%	41.81%	0.04%
Jsteg 50%	0.20%	57.59%	0.00%	0.04%	2.40%	39.58%	0.20%
Jsteg 25%	1.04%	57.63%	0.00%	5.47%	3.67%	30.99%	1.20%
MM2-(1,3,2)	0.56%	0.92%	0.00%	0.80%	91.29%	1.92%	4.51%
MM2-(1,7,3)	0.92%	0.20%	0.00%	14.18%	27.08%	1.68%	55.95%
MM2-(1,15,4)	10.34%	0.44%	0.00%	27.52%	1.24%	0.68%	59.78%
MM3-(1,3,2)	0.44%	1.04%	0.00%	0.84%	91.61%	1.84%	4.23%
MM3-(1,7,3)	1.08%	0.20%	0.00%	15.06%	26.40%	1.88%	55.39%
MM3-(1,15,4)	17.05%	0.44%	0.04%	27.84%	1.16%	0.56%	52.92%
–F5 100%	87.70%	7.23%	0.00%	0.04%	1.28%	3.71%	0.04%
–F5 50%	0.44%	0.08%	0.00%	2.52%	70.41%	1.36%	25.20%
–F5 25%	0.16%	0.00%	0.00%	12.86%	2.76%	0.28%	83.95%

TABLE 5.5. Confusion table of the multi-classifier on images from the testing set embedded by Jsteg, MMx, and –F5. The multi-class detector was not trained to detect any of these algorithms. JP H&S is an abbreviation of JP Hide&Seek.

the cover, F5, and OutGuess classes received the same number of votes, the classifier was programmed to assign the image conservatively to the cover class to have low false positive rate.

5.4. Conclusion

This chapter finished the description of the blind steganalyzer. By presenting results on several different testing sets it was demonstrated that it is possible to reliably classify current popular steganographic algorithms using a multi-classifier based on supervised training. To the best of the author's knowledge, this is the first tool capable of detecting wide range of steganographic algorithms not only in single-compressed, but also in double-compressed JPEG images.

The design of the blind steganalyzer had to overcome several obstacles. Since the compression algorithm of JPEG format is parametrized by the quantization matrix, cover images compressed with different quantization matrices (quality factors) have different statistical properties and the classifier should not be confused by this diversity. Moreover, implementations of some stego programs may under some conditions produce double-compressed images, which further increases the diversity of images to be classified and may lead to very inaccurate steganalysis without appropriate attention. These obstacles were resolved by constructing the classifier from three parts — the double-compression detector, the classifier for single-compressed images, and the classifier for double-compressed images. The task of the double-compression detector is to recognize double-compressed images. Images deemed as double-compressed are sent to primary quality factor estimator and than further to the double-compression multi-classifier designed for two secondary quality factors–75 and 80, the default quality factors of OutGuess and F5. Images detected as single-compressed are sent to single-compression multi-classifier designed for 34 different quality factors.

The classifier was designed under some simplifying assumptions that had to be accepted in order to make the task computationally tractable. First, it was assumed that cover images are always either single-compressed or uncompressed (i.e., cases when the cover image has gone through multiple compression prior to embedding were not considered, even though they are not unlikely). Furthermore, it was assumed that F5 and OutGuess were both run only with quality factors 75 and 80. These assumptions were necessary to reach a reasonable trade-off between the computational cost of training the classifiers and available storage for storing the stego images and their features. Even under these simplifying assumptions, the design and testing of the complete solution required more than 6 million images and over 4TB of disk space. All classifiers were implemented as soft-margin Support Vector Machines with the Gaussian kernel. The feature set for steganalysis was the Merged feature set.

The accuracy of the blind steganalyzer was estimated under various conditions. On images satisfying simplifying assumptions, the accuracy is most of the time above 90% with false alarm rate 1.2%. As the compression history of images departs from simplifying assumptions, the accuracy decreases, which is to be expected.

The ability of the blind steganalyzer to generalize to previously unseen stego methods was evaluated on Jsteg, MMx, and -F5 algorithms. The steganalyzer is able to reliably detect the stego content of novel stego methods, if their embedding mechanism resembles a mechanism of algorithms the steganalyzer was trained on. If the steganalyzer is presented with image with completely new steganographic algorithm, it may fail to detect it, even though the method is otherwise fairly detectable.

CHAPTER 6

Novelty detection in Steganalysis

Universal steganalyzer is a steganalyzer capable of detecting previously unseen (novel) stego methods. It was generally believed that if the steganalyzer is trained on sufficiently many diverse steganographic algorithms, it will become universal. The results obtained from the blind steganalyzer presented in Section 5.3.3 show that in the case of a multi-classifier (classifier classifying into more than two classes), this statement is true only for novel methods resembling some of the methods on which the multi-classifier was trained. When the multi-classifier is presented with a completely different embedding mechanism, it may fail to detect the stego images even for an otherwise fairly easily detectable method, as was the case of the –F5 algorithm.

As was already mentioned in the introduction, it is impossible to build a universal steganalyzer working in the space of all cover images $\mathcal{C}$. Thus, the steganalyzer has to be built in the space with a smaller dimensionality obtained by a projection of $\mathcal{C}$ by means of a feature set $\mathbf{f} : \mathcal{C} \mapsto \mathcal{X} = \mathbb{R}^d$. The essential requirement on the feature set $\mathbf{f}$ is that it has to be *complete* in the sense that

$$D(P_s||P_c) > \epsilon \Rightarrow D(p_s||p_c) > 0,$$

where P_c and P_s are probability distributions on the space $\mathcal{C}$, and p_c, p_s are corresponding induced probability distributions on the projected space $\mathcal{X}$. Universal steganalysis can be described as the composite hypothesis problem

$$\begin{aligned} \mathrm{H}_0 &: \mathbf{x} \sim p_c \\ \mathrm{H}_1 &: \mathbf{x} \nsim p_c. \end{aligned} \tag{6.0.1}$$

The dimensionality of state of the art feature sets aspiring to be complete is usually in order of $10 \sim 10^3$, which is too high to obtain accurate parametric or non-parametric models of p_c. The lack of accurate estimate of p_c prevents the use of tools of detection theory to solve (6.0.1). To alleviate this issue, problem (6.0.1) is transformed to a classification problem. In the context of universal steganalysis, this means that *everything* not resembling a cover image should be classified to the stego class. Such a classifier can be described by the decision function $h : \mathcal{X} \mapsto \{0, 1\}$

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } p_c(\mathbf{x}) > \lambda \\ 0 & \text{otherwise.} \end{cases} \tag{6.0.2}$$

The decision function $h(\mathbf{x})$ partitions the space $\mathcal{X}$ into a region $\mathcal{R}_0$ accepting hypothesis H_0, $\mathcal{R}_0 = \{\mathbf{x}|p_c(\mathbf{x}) > \lambda\}$, and region $\mathcal{R}_1$ rejecting hypothesis H_0 (accepting H_1) $\mathcal{R}_1 = \{\mathbf{x}|p_c(\mathbf{x}) \leq \lambda\}$. The density level λ is the design parameter controlling the trade-off between the probability of false alarm $\alpha = 1 - \int_{\mathcal{R}_0} p_c(\mathbf{x})\mathrm{d}\mathbf{x}$ (cover image classified as stego one) and the probability of missed detection $\beta = \int_{\mathcal{R}_0} p_s(\mathbf{x})\mathrm{d}\mathbf{x}$ (stego image classified as cover).

Notice that the region of acceptance $\mathcal{R}_0$ depends on the portion of pdf $p_c(\mathbf{x})$ above the density level λ and the density level λ, which both are unknown and have to be simultaneously estimated from samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_l\} \sim p_c$. This two-fold estimation makes the design of the universal steganalyzer inherently difficult.

The problem of learning the decision function (6.0.2) only from examples of one class (cover class) is known in machine learning as the novelty / anomaly / density level detection problem. This chapter explores several solutions to this problem [**58, 66, 67, 45**] and discusses their advantages and disadvantages.

6.1. Novelty detection: an overview

6.1.1. One-Class Support Vector Machines (OC-SVM). For a fixed false positive rate α, the problem (6.0.2) can be approached by finding the set C_α (the decision region) with minimum volume, such that the probability $p_c(C_\alpha) \equiv \int_{C_\alpha} p_c(\mathbf{x})\mathrm{d}\mathbf{x} \geq 1 - \alpha$. Denoting the volume of $C \subset \mathcal{X}$ as $\mu(C)$ for some measure $\mu : 2^{\mathcal{X}} \mapsto \mathbb{R}$ (where $2^{\mathcal{X}}$ is the power set of $\mathcal{X}$), for C_α holds

$$C_\alpha = \arg\min_{C \subset \mathcal{X}} \left\{ \mu(C) |\, p_c(C) \geq 1 - \alpha \right\}. \tag{6.1.1}$$

If μ is a Lebesgue measure, then C_α is the smallest set containing at least $1 - \alpha$ fraction of the probability mass. Estimators of this form are called minimum volume estimators.

The size of the power set $2^{\mathcal{X}}$ is usually too large to solve the optimization problem (6.1.1) directly. On the top of it, since the p_c is generally unknown and has to be estimated from finite number of samples, the solution of (6.1.1) can be unstable. Consequently, several relaxations has to be made to make the problem computationally tractable and stable.

One-Class Support Vector Machines (OC-SVM), proposed in [**58**] relaxes (6.1.1) in two ways. First, the minimum is calculated over restricted set $C \in \mathcal{A} \subset 2^{\mathcal{X}}$ consisting of preimages of all half-spaces in $\mathcal{F}$ under a mapping $\phi : \mathcal{X} \mapsto \mathcal{F}$ for some suitably chosen mapping ϕ and space $\mathcal{F}$

$$\mathcal{A} = \left\{ C \subset \mathcal{X} |\, \exists w \in \mathcal{F}, (\mathbf{x} \in C) \Leftrightarrow \langle w, \phi(\mathbf{x}) \rangle_{\mathcal{F}} > 0 \right\}.$$

Second, instead of minimizing the volume of C, which for sets $C \in \mathcal{A}$ might be difficult to calculate and the volume may not be even finite, OC-SVMs minimize an SVM-style regularizer $\mu(C) = \|w\|^2_{\mathcal{F}}$ controlling the length of the weight vector w (and consequently the complexity of the solution) in $\mathcal{F}$. The mapping ϕ as well as the space $\mathcal{F}$ are typically determined from a kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ as $\phi(\mathbf{x}) = k(\cdot, \mathbf{x})$. The space $\mathcal{F}$, obtained by completing the space of all finite linear combinations $\sum a_i \phi(\mathbf{x}_i)$, is a space of functions $\mathcal{X} \mapsto \mathbb{R}$ called Reproducing Kernel Hilbert Space (RKHS). More details about RKHS are presented in Section 8.2.1.

Denoting the training set $\{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$, the training of a OC-SVM leads to a quadratic programming problem [**58**]:

$$\min_{w \in \mathcal{F}, \rho, \xi_i \in \mathbb{R}} \frac{1}{2}\|w\|^2_{\mathcal{F}} + \frac{1}{\nu l} \sum_{i=1}^{l} \xi_i - \rho \tag{6.1.2}$$

subject to

$$\begin{aligned} w \cdot \phi(\mathbf{x}_i) - \rho &\geq -\xi_i,\ i \in \{1, \ldots, l\} \\ \xi_i &\geq 0,\ i \in \{1, \ldots, l\}. \end{aligned}$$

The optimization problem (6.1.2) reveals several interesting facts about OC-SVM. First of all, we can see that the decision function of OC-SVM takes the form

$$h(\mathbf{x}) = \frac{1}{2}(1 + \mathrm{sgn}(w \cdot \phi(\mathbf{x}) - \rho)).$$

More interestingly, the optimization problem does not require all training samples to lie on the correct side of the hyperplane (w, ρ). Notice that if the slack variable $\xi_i > 0$, than the corresponding training sample $\mathbf{x}_i$ is classified as novelty $(w \cdot \phi(\mathbf{x}_i) - \rho < 0)$. The parameter ν controls the trade-off between complexity of the solution and the number of mis-classified points from the training set. Its role is to prevent the optimization reach a degenerate solution because there always exists a combination of (w, ρ) correctly classifying all training samples. If ν is set to the desired false positive rate α, the OC-SVM asymptotically converges to the optimal solution of (6.1.1) [**58**].

The major issue with the use of OC-SVM is setting the parameters of the kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ and the parameter ν controlling the false positive rate. In binary SVMs, this is usually done by estimating the performance by cross-validation on a finite grid of possible parameter values. Since the missed detection rate of OC-SVM cannot be estimated (we have examples only from one class), this approach cannot be used in the context of OC-SVM. To illustrate this issue, one can imagine that it is always possible to choose the kernel wide enough to guarantee a zero false positive rate on testing set. However, the missed detection of this classifier would likely be very high and the lack of stego training examples prevents us to estimate it. The setting of the parameters of OC-SVM thus relies on experience of the user and heuristics. One general heuristics[1] is to use the Gaussian kernel $k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma\|\mathbf{x} - \mathbf{y}\|^2)$ with $\gamma = \frac{1}{\eta^2}$, where η is the median of L_2 distances between samples in the feature space, and setting ν to the desired false positive rate α.

Minimum enclosing balls [**58**] employed by Farid and Lyu [**42**] for blind detection with Wavelet features, can be casted as an OC-SVM. Even though the results presented in [**42**] showed only a minor decrease in detection accuracy with respect to binary SVMs, in experiments presented in Section 6.2 the OC-SVM performed markedly worse. This decrease in performance is most probably due to insufficient number of images in the used database.

6.1.2. One Class Neighbor Machine (OC-NM).

The One Class Neighbor Machine [**45**] relies on a measure capturing "sparsity" of samples in the space. Let $S_l = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ be a set of iid samples drawn according to pdf p_c. The function $M : \mathcal{X} \times S_l \mapsto \mathbb{R}$, defined for all $l \in \mathbb{N}$, is a sparsity measure if and only if $\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$ holds $p_c(\mathbf{x}) > p_c(\mathbf{y}) \Rightarrow M(\mathbf{x}, S_l) < M(\mathbf{y}, S_l)$.

A sparsity measure characterizes closeness of the sample $\mathbf{x}$ to the set of training examples S_l. The rationale behind OC-NMs is to find a threshold ρ so that all samples $\mathbf{x}$ with $M(\mathbf{x}, S_l) > \rho$ are classified as anomalies, i.e., $h(\mathbf{x}) = \mathrm{sgn}(\rho - M(\mathbf{x}, S_l))$.

The training of an OC-NM is very simple, because the only parameter to be learned is the threshold ρ. The training starts with calculating the sparsity of all training samples $m_i = M(\mathbf{x}_i, S_l)$, $i \in \{1, \ldots, l\}$ and ordering them so that $m_1 \le m_2 \le \ldots \le m_l$. By setting $\rho = m_{[(1-\alpha)l]+1}$, it is ensured that at most α fraction of training samples are classified as anomalies. It has been shown that OC-NM converge to optimal solution with increasing number of training samples l [**45**].

Note that there is a key difference between utilizing training samples S_l in OC-NM and in OC-SVM. While OC-SVMs use only a fraction of them during classification (support vectors defining the hyperplane (w, ρ)), OC-NMs use them all, which shows the relation to the nearest neighbor type of classifiers. This difference propagates to the complexity of learning and classification. While the training of OC-SVM is slower than the training of OC-NM, OC-SVM makes the decision faster than OC-NM. Consequently, OC-NM are suitable only for small scale problems.

[1]Private discussion with Bernard Schölkopf.

The original publication [**45**] presents several types of sparsity measures. The one adopted here is based on the Hilbert kernel density estimator

$$M(\mathbf{x}, S_l) = \log\left(\frac{1}{\sum_{i=1}^{l} \frac{1}{\|\mathbf{x}-\mathbf{x}_i\|_2^{hd}}}\right). \tag{6.1.3}$$

The free parameter h controls the smoothness of the measure.

6.1.3. Density Level Detection by Support Vector Machines (DLD-SVM). Intuitively, if at least some information about the distribution of features of stego images p_s is available, the performance of the steganography detector would improve. Steinwart et al. [**66, 67**] introduced an approach to anomaly detection problem that assumes that samples from the pdf of stego images μ are available (hereinafter, μ is used instead of the p_s, because p_s is reserved for the true distribution of stego images). This converts the composite hypothesis testing problem (6.0.2) to a much simpler case

$$\begin{aligned} \mathrm{H}_0 &: \ \mathbf{x}(\mathsf{c}) \sim p_c \\ \mathrm{H}_1 &: \ \mathbf{x}(\mathsf{c}) \sim \mu. \end{aligned} \tag{6.1.4}$$

The pdf μ expresses prior information about the possible location of novelties in the feature space. If no prior information is available, the μ should be chosen to be the least informative, e.g., uniform on $\mathcal{X}$.

The acceptance region of H_0, $\mathcal{R}_0$, is determined by the decision function $f(\mathbf{x}) : \mathcal{X} \mapsto \{-1, +1\}$, $\mathcal{R}_0 = \{\mathbf{x} \in \mathcal{X} | f(\mathbf{x}) = +1\}$ to be learned from available samples (the training set) $\{(\mathbf{x}_1, +1), \ldots, (\mathbf{x}_{\tilde{l}}, +1), (\mathbf{x}_{\tilde{l}+1}, -1), \ldots, (\mathbf{x}_l, -1)\}$, where $\mathbf{x}_1, \ldots, \mathbf{x}_{\tilde{l}} \sim p_c$ and $\mathbf{x}_{\tilde{l}+1}, \ldots, \mathbf{x}_l \sim \mu$. The decision function $f(\mathbf{x})$ can be learned by any method for binary classification. The authors showed that if the probability measure of p_c is absolutely continuous with respect to the probability measure defined by μ, and if the decision function $f(\mathbf{x})$ is implemented by Support Vector Machines, this approach guarantees nearly optimal finite sample performance. In [**66**], DLD-SVMs were compared to other approaches and were reported to perform very well. The issue of DLD-SVM is that under no prior information about μ, the uniform distribution μ does not scale well with the dimensionality of the feature space d. The scalability issue can be illustrated by the following simple example. Let μ be uniform, the dimension of feature space be $d = 300$ and the number of training examples 2×100000 examples (a very optimistic scenario). Under this setting, there are relatively $\log_{300}(10000) \approx 2.01$ examples drawn according to μ per each dimension, which is clearly not enough to learn $f(\mathbf{x})$ with reasonable precision. The examples from μ should "surround" examples from p_c in order to estimate the decision boundary accurately enough.

The only way to remedy this curse of dimensionality of DLD-SVM is to localize the region of possible novelties. In the universal steganalysis this can be achieved, by training a cover vs. all-stego classifier on examples of cover and stego images embedded by some "known" algorithms. The hope here is that if the classifier is trained on sufficiently diverse set of algorithms, it may be able to detect new steganographic schemes. Unfortunately as will be shown later, if the cover vs. all-stego classifier is presented with stego algorithm not resembling any stego mechanism used during training, it may dreadfully fail. On the other hand, this approach offers a very good detection accuracy on "known" algorithms, which is important if the steganography detector is used as a pre-classifier for a multi-class detector assigning images to known steganographic algorithms.

6.2. Experimental comparison

This section presents experimental comparison of the novelty detection methods described in the previous section. The training and testing conditions were similar to the conditions under which the blind steganalyzer was developed. All images involved in this experiment were single-compressed JPEG images with quality factor 75. Images were divided according to the original raw image into training set containing 3500 raw images and testing set containing 2506 raw images. The stego algorithms were divided into known algorithms: F5, JP Hide&Seek, MBS1, MBS2, Steghide, and OutGuess, and unknown algorithms –F5, MM2, MM3, and F5 without shrinkage (nsF5) [**21**]. While the known algorithms could be possibly used during training of the classifiers, the unknown algorithms had to be kept unknown because they are needed to estimate the ability of the detectors to detect novel algorithms. All classifiers used the Merged feature set (Chapter 4). The parameters of methods were set either according to heuristics (OC-SVM, OC-NM) or by grid-search (DLD-SVM) to the target 1% false positive rate.

For a OC-SVM, the heuristics described above was used. The width of the Gaussian kernel was set to $\gamma = 0.181526$ according to the "median" rule, and $\nu = 0.01$, which is the desired false positive rate. The data was preprocessed by scaling so that all features of the testing set were in the range $[-1, +1]$ (the scaling parameters were derived from cover images only).

Although several different sparsity measures for OC-NM were proposed in [**45**], only the results obtained from the best performing one (6.1.3) are presented here. This measure was used with the following values of the parameter $h = \{0.01, 0.02, 0.05, 0.08, 0.1\}$ based on the recommendations in the paper. The detection accuracy varied very little with h. The results shown in Tables 6.1 and 6.2 were obtained for $h = 0.01$.

The data pre-processing in DLD-SVM is not so straightforward, because the samples from the underlying distribution μ have to be generated, which was done in the following manner. First, the scaling parameters on 3400 examples of cover images bringing all features to the range $[-1, +1]$ were derived. Then, 15000 artificial samples were generated according to the underlying pdf $\mu = U([-1, +1]^d)$. Because the resulting training set with 18000 examples was imbalanced (there were more examples from one of the classes), weighted Support Vector Machines (2C-SVM) with Gaussian kernel were employed. The hyper-parameters (C^+, C^-, γ) were determined by the combination of grid-search and 5-fold cross-validation, as described in Appendix A.4. As expected, all triplets (C^+, C^-, γ) evaluated during grid-search had false negative rate (class μ detected as cover) always equal to 0. This shows that not enough samples from μ have been provided. Even though it is easy to generate more samples from μ, the problem becomes quickly computationally intractable, since the complexity of training a SVM is approximately $O(l^3)$, where l is number of training examples. Nevertheless, for the sake of completeness the results of this approach are presented under the label "$\mathrm{DLD} - \mathrm{SVM}_{\mathrm{uni}}$"

In order to localize the novelties in the input space, a binary C-SVM with Gaussian kernel was trained on 3400 examples of cover images and 3400 examples of images embedded by "known" algorithms with message lengths $100\%, 50\%$, and 25% of their capacity (the only exception were images from MBS2 that were embedded with 30% of capacity of MBS1). As in the case of the $\mathrm{DLD} - \mathrm{SVM}_{\mathrm{uni}}$, the hyper-parameters C and γ were determined by a grid-search combined with 5-fold cross-validation. This approach, further denoted as "$\mathrm{DLD} - \mathrm{SVM}_{\mathrm{loc}}$", is a practical embodiment of a cover vs. all-stego binary classifier.

The accuracy of detectors was estimated on JPEG images created from 2504 raw images not used during training. Images were embedded with messages with length 100%,75%, 50%, 25%, 20%, 10%, and 5% bits per non zero AC coefficient (bpac) by –F5, Jsteg and nsF5,

Target	OC-SVM	OC-SVM$_{\mathrm{shift}}$	OC-NM	DLD-SVM$_{\mathrm{uni}}$	DLD-SVM$_{\mathrm{loc}}$
–F5 100%	100%	100%	100.00%	98.88%	99.08%
–F5 75%	100%	100%	100.00%	89.50%	99.44%
–F5 50%	100%	100%	100.00%	6.75%	99.60%
–F5 25%	100%	95.64%	93.93%	0.12%	98.48%
–F5 20%	99.6%	66.57%	55.87%	0.16%	96.09%
–F5 10%	17.73%	3.7%	3.27%	0.16%	33.11%
–F5 5%	6.70%	1.55%	1.48%	0.12%	3.55%
nsF5 100%	100%	100%	99.96%	16.41%	99.96%
nsF5 75%	100%	99.96%	99.92%	3.04%	99.96%
nsF5 50%	98.76%	74.56%	80.91%	0.20%	99.72%
nsF5 25%	11.50%	2.87%	3.19%	0.12%	88.86%
nsF5 20%	9.78%	2.07%	2.24%	0.12%	72.12%
nsF5 10%	5.99%	1.47%	1.44%	0.12%	6.11%
nsF5 5%	5.47%	1.31%	1.40%	0.12%	1.72%
MM2-(1,3,2)	100%	100%	100.00%	18.37%	99.64%
MM2-(1,7,3)	100%	100%	99.92%	0.12%	99.20%
MM2-(1,15,4)	62.61%	20.24%	17.69%	0.12%	53.67%
MM3-(1,3,2)	100%	100%	100.00%	18.29%	99.72%
MM3-(1,7,3)	100%	100%	99.92%	0.12%	99.32%
MM3-(1,15,4)	51.71%	17.17%	15.14%	0.12%	58.51%
Jsteg 100%	100%	100%	100%	98.24%	42.41%
Jsteg 75%	100%	100%	100%	87.85%	42.33%
Jsteg 50%	100%	100%	100%	66.85%	42.37%
Jsteg 40%	100%	100%	100%	60.54%	42.29%
Jsteg 25%	100%	99.84%	99.45%	56.94%	42.05%
Jsteg 20%	99.88%	99.12%	98.09%	56.78%	42.09%
Jsteg 10%	96.13%	83.11%	65.36%	56.62%	32.98%
Jsteg 5%	78.87%	63.53%	40.27%	56.62%	5.99%
Cover	94.76%	98.64%	98.64%	99.88%	98.96%

TABLE 6.1. Comparison of accuracy of general steganography detectors on "unknown" algorithms. The detector "OC-SVM$_{\mathrm{shift}}$" is an OC-SVM classifier with the threshold shifted to match the false positive rate of the OC-NM classifier.

and with messages with length 66%, 42%, and 26% bpac by MM2 and MM3 (the message lengths for MMx correspond to the maximal messages for Hamming codes (1,3,2), (1,7,3), and (1,15,4)).

6.2.1. Accuracy on stego images. Not surprisingly, the DLD-SVM$_{\mathrm{loc}}$ detector performs the best on all known algorithms and all unknown algorithms with the exception of Jsteg, where it grossly fails. The failure on Jsteg is rather surprising considering the fact that the blind steganalyzer and DLD-SVM$_{\mathrm{loc}}$ were constructed under similar conditions and Jsteg was easily detectable by the blind steganalyzer (Section 5.3.3). This experiment shows that DLD-SVM$_{\mathrm{loc}}$ suffers from the same drawback as the blind steganalyzer—it may fail to detect stego algorithms with a completely different embedding mechanism. In contrast, all true novelty detection methods (OC-SVM, OC-NM, and DLD-SVM$_{\mathrm{uni}}$) detected Jsteg even at low embedding rates.

Target	OC-SVM	OC-SVM$_{\text{shift}}$	OC-NM	DLD-SVM$_{\text{uni}}$	DLD-SVM$_{\text{loc}}$
F5 100%	100.00%	99.60%	98.96%	1.92%	99.96%
F5 50%	78.11%	29.09%	20.10%	0.12%	99.60%
F5 25%	13.06%	2.64%	2.40%	0.12%	90.73%
JP Hide&Seek 100%	100%	99.68%	99.52%	0.52%	99.84%
JP Hide&Seek 50%	85.18%	54.19%	41.73%	0.48%	98.28%
JP Hide&Seek 25%	40.13	21.60%	19.04%	0.44%	73.52%
MBS1 100%	100.00%	100.00%	99.92%	0.20%	99.96%
MBS1 50%	97.88%	53.36%	29.50%	0.16%	99.80%
MBS1 30%	35.12%	7.03%	4.27%	0.12%	98.88%
MBS1 25%	21.33%	3.59%	2.56%	0.12%	96.81%
MBS1 15%	9.95%	1.84%	1.76%	0.12%	71.19%
MBS2 30%	93.49%	55.95%	32.47%	0.12%	99.12%
MBS2 15%	33.63%	5.51%	2.88%	0.12%	77.92%
OutGuess 100%	100.00%	100.00%	100.00%	1.72%	99.96%
OutGuess 50%	99.80%	80.07%	57.51%	0.20%	99.96%
OutGuess 25%	41.25%	8.15%	5.19%	0.12%	98.12%
Steghide 100%	100.00%	99.96%	99.44%	0.20%	99.96%
Steghide 50%	85.90%	27.76%	16.61%	0.16%	99.84%
Steghide 25%	18.61%	4.19%	2.84%	0.20%	96.37%

TABLE 6.2. Comparison of accuracy of general steganography detectors on "known" algorithms. Note that except for the detector DLD-SVM$_{\text{loc}}$, these algorithms were not used to create images in the training set, which makes them "unknown." The detector "OC-SVM$_{\text{shift}}$" is an OC-SVM classifier with the threshold shifted to match the false positive rate of the OC-NM classifier.

In order to compare OC-SVM, OC-NM, and DLD-SVM$_{\text{uni}}$ more fairly, the threshold of OC-SVM was shifted so that the false positive rate of OC-SVM and OC-NM on testing images was the same. The performance of this shifted OC-SVM (labeled in Table 6.1 as "OC-SVM$_{\text{shift}}$") is better than the performance of OC-NM, especially on "known" algorithms (Table 6.2).

Table 6.1 shows that the –F5 algorithm, which was not detected by the blind steganalyzer in Section 5.3.3, is now reliably detected by all classifiers except DLD-SVM$_{\text{uni}}$, which detected it only when the images were embedded with 75% or larger payload.

As expected, the DLD-SVM$_{\text{uni}}$ method performed the worst because it suffers from curse of dimensionality. It can only detect poor algorithms, such as –F5 or Jsteg.

The results presented in Tables 6.1 and 6.2 also reveal differences between novelty and binary detectors. DLD-SVM$_{\text{loc}}$ (and all binary classifiers in general) identify the boundary between cover and stego images only in those parts of the feature space that are occupied by the features from the known stego methods. In those regions of the feature space where the examples from the stego class are absent, the decision boundary can be too far from the cover class making the classifier vulnerable to catastrophic failures to detect stego images falling into this region. By contrast, novelty detectors try to identify the decision boundary in all parts of the feature space, which makes them suitable for universal steganalysis.

The comparison of different universal steganography detectors shows that the choice has to be made with respect to the intended application. If the detector is going to be used as a pre-classifier module in a multi-class detector, the DLD-SVM$_{\text{loc}}$ (cover vs. all-stego) is a good choice because of its superior performance on "known" algorithms in comparison

Target	OC-SVM	OC-SVM$_{\text{shift}}$	OC-NM	DLD-SVM$_{\text{uni}}$	DLD-SVM$_{\text{loc}}$
Blurring $\sigma = 0.4$	94.33%	98.48%	98.92%	99.88%	98.84%
Blurring $\sigma = 0.8$	93.97%	98.32%	98.76%	99.80%	98.84%
Blurring $\sigma = 1.2$	91.85%	98.00%	98.72%	99.76%	98.76%
Blurring $\sigma = 1.6$	88.06%	97.68%	98.52%	99.80%	98.44%
Blurring $\sigma = 2.0$	79.03%	96.92%	98.08%	99.80%	98.04%
Color quantization	93.13%	98.72%	99.00%	99.88%	97.60%
Despeckling	93.05%	98.08%	98.68%	99.76%	98.64%
Gamma corr. $\gamma = 0.7$	95.21%	98.52%	98.88%	99.84%	98.64%
Normalization	94.97%	99.04%	99.44%	100.00%	99.60%
Sharpened $\sigma = 0.4$	94.81%	98.72%	98.52%	99.88%	98.84%
No processing	94.76%	98.64%	98.64%	99.88%	98.96%

TABLE 6.3. Percentage of processed covers detected correctly as covers.

PQF	OC-SVM	OC-SVM$_{\text{shift}}$	OC-NM	DLD-SVM$_{\text{uni}}$	DLD-SVM$_{\text{loc}}$
65	0.00%	0.00%	0.00%	4.67%	0.04%
70	0.00%	0.08%	0.28%	95.61%	0.12%
80	0.00%	0.00%	0.32%	99.32%	99.84%
85	0.00%	0.00%	0.08%	92.93%	99.84%
90	6.67%	27.76%	38.54%	99.96%	32.15%

TABLE 6.4. Percentage of correctly classified covers that were double-compressed using primary quality factor PQF and secondary quality factor 75.

to other approaches (see Table 6.2). For a universal blind detector trained only on cover images, the OC-SVM offers slightly better performance than OC-NM, though the tricky setting of hyper-parameters makes them difficult to implement in practice.

6.2.2. Steganalysis of processed cover images. It has been recognized by the research community that the source of covers has a major influence on steganalysis in the spatial domain. Steganalysis of JPEG covers is generally expected to be less sensitive to the cover source due to the quantization performed during JPEG compression. The one-class novelty detectors described in the previous section, however, may be more sensitive to the cover source because they are only trained on covers. In this section, the performance of the universal steganalyzers on images that underwent various processing is verified in order to see if covers processed by common image processing operations are likely to be mistaken for stego images.

To this end, the testing database of 2504 images was processed using the following operations: blurring with Gaussian kernel with kernel width $\sigma \in \{0.4, 0.8, 1.2, 1.6, 2.0\}$, sharpening with $\sigma \in \{0.4, 0.8, 1.2, 1.6, 2.0\}$, despeckling, color quantization to 256 colors, histogram normalization in all 3 color channels, and gamma correction with $\gamma \in \{0.7, 0.9, 1.1, 1.3\}$. All operations were carried out in Image Magick's Convert routine. To avoid producing double compressed JPEGs, the source image was always in raw format and after the processing the image was saved it as 75% quality JPEG.

Table 6.3 shows the percentage of correctly classified processed covers by all five tested steganalyzers. Blurring with Gaussian kernel with $\sigma = 1.6$ $\sigma = 2.0$ increased the false positive rate the most, especially for OC-SVM. The other processing did not have a significant influence on the detection accuracy.

Because the Merged features are computed directly from quantized DCT coefficients, they are very sensitive to repetitive JPEG compression. A double-compressed cover image exhibit artifacts due to double quantization of DCT coefficients, which is likely to be misinterpreted by the one-class detectors as an anomalous image. Table 6.4 confirms this educated guess. It shows the percentage of correctly classified covers that were double JPEG compressed with the primary quality factor $\text{PQF} \in \{65, 70, 80, 85, 90\}$ and secondary quality factor 75. The negative influence of double compression on steganalysis that uses features computed from DCT coefficients is well-known. The blind steganalyzer (Chapter 5) deals with double-compressed images by having modules to detect them 3.2.2, estimate their primary quantization matrix 3.2.3.1, and by training appropriate detectors for double-compressed JPEG images 5.2.

Overall the OC-NM is more robust to processing than $\text{OC-SVM}_{\text{shift}}$ (see Tables 6.3 and 6.4). On the other hand, as reported in Section 6.2, $\text{OC-SVM}_{\text{shift}}$ better detects stego content than OC-NM. This indicates that the decision boundary of $\text{OC-SVM}_{\text{shift}}$ surrounds cover images more tightly.

Binary classifiers ($\text{DLD-SVM}_{\text{uni}}$ and $\text{DLD-SVM}_{\text{loc}}$) are less likely to mis-classify processed images (especially double-compressed images) than novelty detectors.

6.3. Conclusions

A detector trained to recognize variety of steganographic algorithms does not necessarily have to be a good universal steganography detector because it can fail to recognize images produced by steganographic methods with a completely novel embedding mechanism as stego. This applies to both multi-class detectors and binary cover-against-all-stego detectors.

In this chapter several existing approaches to anomaly detection were adapted to steganalysis, and their performance was compared. The methods differed in their machine learning techniques as well as in utilizing side information in the form of examples of "known" steganographic algorithms. Among the techniques that do not utilize any information about stego images, the one-class SVM trained only on examples of cover images had the best overall performance and was less prone to failures to detect an unknown stego method. The detection accuracy of one-class detectors on known stego algorithms is understandably somewhat worse than detection accuracy of binary cover-against-all-stego detectors trained on such stego images.

For applications where reliable universal blind detector is required, such as for automatic traffic monitoring, targeted steganalyzers or multi-class detectors should be supplemented with a reliable one-class detector.

CHAPTER 7

Alternative use of blind steganalysis feature sets

The combination of machine-learning algorithms and features for blind steganalysis does not need to be limited to blind steganalysis (composite hypothesis test 1.3.1). This chapter shows some applications of the combination of machine learning tools and feature extraction in steganography and steganalysis.

7.1. Targeted steganalysis

By training a binary classifier (for example, a soft-margin Support Vector Machine, Appendix A.2) on training set consisting only from examples of cover images and images embedded by a specific steganographic algorithm, Eve obtains a targeted steganalyzer. Naturally, this targeted steganalyzer naturally has better performance than the blind steganalyzer, because its construction utilized side information about the steganographic method used by Alice and Bob. Moreover, if a targeted attack on the scheme exists, Eve can further improve accuracy of her steganalyzer by augmenting the general feature set with features used in the targeted attack. This feature enhancement is always recommended, since the improvement in the accuracy can be significant. This approach to targeted steganalysis often produces the most reliable steganalysis.

In order to show how the side information can improve the accuracy, Table 7.1 shows the accuracy of targeted steganalyzers for single-compressed JPEG images with quality

		stego images with message length			
cover vs.	cover	25%	30%	50%	100%
F5	99.64%	98.36%	—	99.84%	99.92%
JP Hide&Seek	99.56%	92.01%	—	99.60%	99.52%
JSteg	99.92%	98.40%	—	99.20%	99.96%
MBS1	99.88%	99.72%	—	99.92%	99.96%
MBS2	99.92%	—	100.00%	—	—
nsF5	98.84%	97.24%	—	99.84%	99.84%
OutGuess	99.76%	99.48%	—	99.96%	100.00%
Steghide	99.92%	99.32%	—	99.92%	100.00%
–F5	99.96%	99.72%	—	99.96%	99.72%

		stego images with message length			
cover vs.	cover	26%	—	42%	66%
MM2	99.96%	99.40%	—	99.92%	99.96%
MM3	99.80%	99.40%	—	99.92%	99.96%

TABLE 7.1. Detection accuracy of targeted steganalyzers calculated on images from testing set. All steganalyzers are soft-margin C-SVMs. The message length of stego images embedded by MMx correspond to the maximal messages for Hamming codes (1,3,2), (1,7,3), and (1,15,4)).

factor 75 detecting one of 10 steganographic algorithms: –F5, F5, JP Hide&Seek, MBS1, MBS2, MM2, MM3, nsF5, Steghide, and OutGuess. All targeted steganalyzers were implemented as C-SVMs trained on 3400 cover examples and 3400 stego examples with an even mixture of available message lengths. The selection of the hyper-parameters was done by grid-search on an unbounded grid combined with 5-fold cross-validation, as described in Appendix A.4. Features were pre-processed by scaling to bring the individual features to the interval $[-1, +1]$.

By comparing the detection accuracy of targeted classifiers (Table 7.1) to detection accuracies of universal steganalyzers (OC-SVM, OC-SVM shift, and OC-NM classifiers in Tables 6.1 and 6.2), it is obvious that the targeted classifiers are significantly more accurate. This is especially true for images containing short messages, where the distortion caused by steganography is more subtle. For example, images embedded by nsF5 with messages of length 25% of the image capacity[1] are almost undetectable by universal steganalyzers, while the targeted steganalyzer detects them reliably.

The conclusion of this experiment is not surprising. The decision boundary between cover and stego class identified by one-class classifiers has to be shifted more towards stego class, because one-class classifiers play the "safe game" in order to achieve low probability of false alarms. Examples from the stego class available to the targeted steganalyzer allow locating the boundary separating the features of cover and stego images more precisely, which is turns into higher accuracy on stego images.

The lesson learned from this experiment can be extended to other forms of side information. If any side information about the stego channel is available, it should be always utilized in order to improve the accuracy of the steganalyzer.

7.2. Dimensionality reduction

As mentioned in the introduction, targeted steganalysis often gives excellent results with only one feature. An easy method to obtain a single feature is to use the unthresholded output of soft-margin Support Vector Machine. Using notation of Appendix A, this feature can be written as

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_j) - b.$$

It is important to distinguish feature selection from dimensionality reduction. Denoting the complete feature set $\mathbf{f} : \mathcal{C} \mapsto \mathcal{X} \in \mathbb{R}^d$, *Dimensionality reduction* aims to find a function $\mathbf{g} : \mathbb{R}^d \mapsto \mathbb{R}^n, n < d$, so that $\mathbf{g} \circ \mathbf{f}$ preserves important characteristics of the data. A typical example of dimensionality reduction is the Principal Component Analysis or the Independent Component Analysis [**29**]. *Feature selection* can be understood as a special case of dimensionality reduction, where $\mathbf{g}$ takes the form of a product of a binary diagonal matrix with the original vector, i.e., $\{0, 1\}^{d,d}\mathbf{x}$. Feature selection attempts to identify a subset of features $\{\mathbf{f}_{i_1}, \ldots, \mathbf{f}_{i_n}\} \in \mathbb{R}^n$, $i_j \in \{1, \ldots, d\}$, $n < d$, so that the detection performance of the feature subset $\{\mathbf{f}_{i_1}, \ldots, \mathbf{f}_{i_n}\}$ improves or at least not significantly worse that the detection performance of the whole feature set (sometimes minor decrease in the detection performance is acceptable if the dimensionality reduction is significant).

There are several reasons why Eve should be interested in feature selection. Non-informative features can significantly decrease the detection accuracy of the classifier. A simple toy example showing this phenomenon for Support Vector Machines can be found in [**73**]. Additionally, Identification of features contributing the most to the detection can highlight weaknesses of the steganographic scheme. This knowledge can be utilized in

[1]The capacity of the image for embedding by nsF5 is equal to number of non-zero AC coefficients.

forensic analysis and for design of new stego schemes. Finally, a smaller number of features reduces the complexity of the steganalyzer.

The pioneering work of feature selection in the field of steganalysis is due to Miche et al. [**43**]. The authors identified features within the original 23 DCT features (Chapter 4.2) important for detection of F5, OutGuess, and Steghide algorithms. The results are quite surprising, since only 14 (13) out of 23 features are needed for the classification of Outguess (Steghide) with the same accuracy as is achieved with the full set of features. On the other hand, to detect F5 algorithm, 22 out of 23 features are needed, which shows that a F5 is perhaps more advanced than Outguess or Steghide.

Dimensionality reduction methods such as the Principal Component Analysis (PCA) or Independent Component Analysis (ICA) were not yet widely used in steganography and steganalysis. In this dissertation, PCA was used in Chapter 4 to show the clusters formed by features of different steganographic schemes.

7.3. Practical benchmarking of steganographic schemes

As was already mentioned in the introduction, the feature set for the blind steganalysis capable of detecting a wide range of steganographic algorithms provides a good low-dimensional model of cover images. This is an important characteristic of feature sets used to verify and compare security of steganographic schemes.

From the information theoretic point of view, the best measure comparing the distributions of cover, p_c, and stego, p_s, would be the KL divergence, as it provides the upper bound on the best detector Eve can build. However the lack of parametric models for p_c and p_s together with the high dimension of feature spaces aspiring to be complete (usually $10 \sim 10^3$ dimensions) prevents estimating p_c and p_s with sufficient precision to calculate the KL divergence. (If a reasonable number of samples ($\sim 10^5$) is assumed) To remedy these issues, sub-optimal measures, such as the classification error of SVMs or Maximum Mean Discrepancy advocated in the second part of the dissertation, may be used.

Since the second part of this dissertation is entirely dedicated to the issue of steganographic benchmarking, this topic is not discussed here further.

7.4. Steganography design

All three applications of feature sets and pattern recognition tools presented above can be utilized in design of steganographic algorithm.

The security of a new algorithm can be readily verified with respect to the feature set. In fact, this approach to steganography has become standard today and majority of research articles proposing new embedding algorithms report steganalysis results using a blind steganalyzer. For example in [**63**] by Solanki et al., where the YASS steganographic scheme for JPEG images was proposed, the security of the YASS algorithm was verified with respect to 6 different feature sets. The same paper also compared the security of the proposed scheme to other algorithms (for a fixed message length).

Changes in steganographic features can be used as a distortion measure during embedding. This idea was fully exploited in the Feature Correction Method (FCM) proposed in [**38**]. FCM can be understood as an extension of Minimal Distortion Steganography introduced by Kim et al. [**37**]. When embedding a message bit(s), several different ways of making the embedding change encoding the bit(s) are explored, and the one causing the smallest distortion is used. FCM method calculates the distortion measure by use of the Merged features **f** as

$$\|\mathbf{f}(J) - \mathbf{f}(J^{'})\|_W, \tag{7.4.1}$$

where J denotes the original cover image, $J^{'}$ denotes the modified image, and $\| \cdot \|_W$ is a weighted norm. FCM also saves a portion of the image for future feature restoration, as, for example, OutGuess and MBS2 do. After embedding, FCM makes changes in the untouched portion of the image to minimize the distortion measured by (7.2), in order to bring the feature vector of the stego image back to the feature vector of the cover image $\mathbf{f}(J)$.

There is a potential danger in designing the steganographic scheme to be undetectable by a fixed feature set. The scheme can be potentially very detectable by other feature sets, or even by slight change of the original feature set (for example using a different cropping in the calibration) [**38**].

Part 2

Security of steganographic schemes

CHAPTER 8

Revisited Security of Steganographic Scheme

The measure of steganographic security in Cachin's definition (see Section 1.1.1 and [**9**]) is the Kullback–Leibler divergence

$$D_{\mathrm{KL}}(P_c||P_s) = \sum_{c\in\mathcal{C}} P_c(c)\log\frac{P_c(c)}{P_s(c)}$$

between the probability distribution of covers P_c and stego objects P_s on the space of all cover objects $\mathcal{C}$. Therefore, it would make sense to evaluate and compare security of steganographic schemes directly by the value $D_{\mathrm{KL}}(P_c||P_s)$. As was already mentioned in the introduction, it is impossible to calculate $D_{\mathrm{KL}}(P_c||P_s)$ directly in $\mathcal{C}$ due to its high dimensionality. The need to compare security of steganographic algorithms prompted the steganographic community to adopt alternative benchmarks. Most of the time, the comparison is done by reporting detection accuracy of (targeted) classifiers trained to detect a particular steganographic scheme [**21, 36, 10**] (see Chapter 7.1 for examples of targeted steganalyzers). Results of such benchmarks depend on several design choices: feature set, set of cover images, set of stego images, classifier, and a functional assigning a single number to the ROC curve of the classifier. In the rest of this section, these design choices are discussed in more detail in order to show their influence on the final benchmark.

8.0.0.1. *Feature set.* The purpose of the feature [**51, 17, 3, 2, 74, 61, 42, 16**] set is to reduce the dimensionality of the space of all cover objects $\mathcal{C}$. An alternative way to do so is to model the space $\mathcal{C}$ analytically, for example as a sequence of iid random variables [**44**] or by Markov chains [**64**]. The major advantage of analytical schemes is that by using them it might be possible for a fixed steganographic scheme to express the KL divergence $D_{\mathrm{KL}}(P_c\|P_s(\alpha))$ as a function of the embedding rate α. Unfortunately, analytical models are not mature enough to capture complex cover objects, such as digital images. It happened in the past that "provably secure" steganographic schemes were easily broken by using a better model of cover objects and designing appropriate test statistics. An example of such successful attacks on steganographic systems that preserve first order statistics of DCT coefficients in JPEG images [**55, 46, 27, 14, 64**] can be found in [**51**].

Since the available analytical models proved to be insufficient to truthfully describe natural images, the steganalytic feature set $\mathbf{f} : \mathcal{C} \mapsto \mathcal{X}$ has to be used to reduce the dimension of $\mathcal{C}$ (in steganalysis, the model space $\mathcal{X}$ is almost exclusively the Euclidean space $\mathcal{X} = \mathbb{R}^d$, $d \in \mathbb{N}$). The choice of the feature set for benchmarking is crucial, since the steganographic security / benchmark has to be defined *with respect* to the feature set. The feature set chosen for the benchmark should be complete in the sensse that for every steganographic scheme should hold

$$D_{\mathrm{KL}}(P_c\|P_s) > \epsilon \Rightarrow D_{\mathrm{KL}}(p_c\|p_s) > 0,$$

where p_c, p_s denote the probability distribution of cover and stego objects, respectively on $\mathcal{X}$. Complete feature sets are hard to find. Therefore, in practice the requirement of completeness is relaxed to a weaker property, namely to the requirement that it has to be hard to practically construct a stego scheme such that $D_{\mathrm{KL}}(p_s\|p_c) = 0$.

State of the art feature sets are not complete. An example how one can design a stego scheme hard to detect by a given features set is shown in [**38**]. Consequently, it can happen that steganographic schemes secure with respect to one feature set can be easily detectable by different feature set. Authors in [**38**] designed the stego scheme to be undetectable by the Merged feature set (Section 4.4), but the same stego-scheme becomes *very detectable* just by changing the cropping factor in calibration (Section 4.1).

8.0.0.2. *Cover images.* Since benchmarking / verification of security of steganographic schemes is performed on the finite set of images, the choice of images plays an important role. While it is natural to require images in the set to be as diverse as possible (the influence of cover images was already discussed in Section 1.3.5), it is important to realize the influence of JPEG compression on benchmarking JPEG stego-schemes. The results of the blind steganalyzer and one-class classifiers presented in Sections 5.3 and 6.2.2 showed that repeated JPEG compression has a major impact on the detection performance of steganalytic schemes calculating features directly from the embedding domain (DCT domain for JPEG images). There are two fundamentally different ways how to approach the construction of the cover set.

In [**36**], where a comparison of steganographic security of several algorithms together with comparison of detection accuracy of several feature sets was presented, a web crawling robot was employed to create a database of approx. 1.1 million JPEG images. Since this set contained images highly likely to be encountered in practice, calculated accuracies provide a good estimate that can be expected in real use. On the other hand, since there was virtually no control over the compression history of the images, images produced by different algorithms might be incomparable. For example, images embedded by Outguess underwent additional JPEG compression during embedding, which can significantly alter statistics of images and make them more detectable. Moreover, feature sets targeted for JPEG images were applied naïvely (no compensation for double-compression or specialization of the classifiers for a quality factor was used), which means that their steganalytic power was not fully utilized. It is important to take care of these nuances, since they can significantly skew the results of the benchmark.

A completely different approach was employed in [**21**], where the security of 10 steganographic algorithms was compared. JPEGs used in the benchmark were prepared from raw images in such a way that all resulting images were single-compressed JPEGs with quality factor 70. Comparison on this data set does not suffer from unexpected effects explained above. It truly compares the security of algorithms with respect to given feature set, which is utilized to its maximum. Such a comparison can reveal weaknesses of individual algorithms. This approach is adopted in this dissertation in Section 8.3.2.

8.0.0.3. *Stego images.* The aim of the steganographic benchmark is to evaluate which steganographic scheme is more secure (less detectable). However, the outcome depends on how the steganography is used. It can happen that one scheme can be more detectable than other on one payload size and less detectable on a different payload size. This is especially likely, if one of benchmarked algorithms uses matrix embedding [**18**], which exhibits sharp non-linear decrease in detectability with decreasing payload due to significantly lower number of embedding changes. Some methods do not allow matrix embedding (e.g., adaptive schemes). Moreover, some steganographic algorithms are inherently limited to binary codes, such as methods based on perturbed quantization [**37, 20**], while other methods that use ± 1 type of embedding can utilize more powerful ternary codes [**18**]. The implication of these differences is that some steganographic methods can embed significantly higher payload than other method for the same distortion budget (number of changes performed during embedding). Fixing the distortion budget instead of the payload would, however, benchmark the type of embedding operation rather than the whole scheme.

A tempting solution would be to somehow simulate real-life usage. To do so, a statistical distribution of payloads typically used needs to be known. Since there is nothing that can be assumed about this prior distribution, this solution is not plausible.

Taking into account the issues described above, it seems that a reasonable option is to fix the message length with respect to the number of coefficients in the image usable for steganography. For JPEG images, the *embedding rate* α (also called *relative payload*), is defined as the ratio between the message length in bits and the number of non zero AC coefficients in the cover JPEG image (bpac). Thus, for each particular image, every stego method embeds the same relative payload. This methodology for setting the message length in the benchmark was used in [**21**]. The major advantage is that by fixing $\epsilon > 0$ in the definition of steganographic security, the benchmark allows to state that a certain steganographic method becomes ϵ-secure at relative payload $\alpha(\epsilon)$. Fixing the relative message length also makes intuitive sense because people might subconsciously use a bigger cover for large messages and a smaller cover for short messages. Similar approach was used in [**36**], where authors defined the relative embedding rate with respect to non zero DCT coefficients.

8.0.0.4. *Machine learning engine.* The machine learning / statistical tool used to implement the decision function $h : \mathcal{X} \mapsto \{0, 1\}$,

$$h(\mathbf{x}) = \begin{cases} 0 & \mathbf{x} \sim p_c \\ 1 & \mathbf{x} \sim p_s, \end{cases}$$

is an important factor of the benchmark. The most popular machine learning tools used in steganography are Fisher Linear Discriminant (FLD) (for example [**17**]) and Support Vector Machines with linear (for example [**36**]) or Gaussian (for example [**21**]) kernel. While FLD and SVM with linear kernel provide somewhat similar results, SVMs with Gaussian kernel are more powerful. Steganographic schemes not detectable with FLD can become detectable with SVM with Gaussian kernel, despite the fact that the same feature set is used.

8.0.0.5. *Operating point of the machine learning engine.* The steganographic benchmark strongly depends on the functional assigning a single value to the ROC curve of the classifier, because ROC curves frequently intersect. Appendix A.4.5.2 reviews several popular functionals in steganography (total minimal decision error [**21, 63**], probability of detection for fixed false alarm rate [**42**], or false alarm for probability of detection 50% [**32**], detection accuracy [**17**], etc.). The problem of assigning a single value to ROC curve is discussed in detail by Ker in [**34**]. For the purpose of benchmarking steganalytic schemes, Ker advocates to use the limit

$$Q = \lim_{\lambda \to 0} \frac{D_{\mathrm{KL}}(f_c \| f_s(\lambda))}{\lambda^2}, \tag{8.0.2}$$

where f_c, $f_s(\lambda)$ is an output of the targeted classifier before thresholding on cover and stego images with change rate λ (number of coefficients changed in the image during embedding) respectively. The reasoning behind a scalar value Q is that over multiple uses of the stego channel, the relative change rate λ must converge to zero to avoid detection. Because for statistically detectable stego schemes the KL divergence is quadratic in λ, $D \approx Q\lambda^2$ as $\lambda \to 0$, the constant Q always exists and shows how quickly the KL divergence of a given steganalysis detector goes to zero. Also notice that Ker's benchmark (8.0.2) removes the problem of choosing the embedding rate of stego images discussed above.

It would be tempting to adopt (8.0.2) in the benchmarking of steganographic schemes. To do so, the change rate λ has to be substituted by the payload α (relative number of bits embedded into the image). After this substitution, KL divergence may in some cases become non-quadratic in payload α around zero. For example, for matrix embedding utilizing optimal codes $\lambda = H^{-1}(\alpha)$ and $D_{\mathrm{KL}} \sim \big(H^{-1}(\alpha)\big)^2$.[1] Although this observation does not preclude the possibility to benchmark steganography in the limit $\alpha \to 0$, this approach is not investigated further.

8.0.1. Ingredients of optimal benchmark. Among the above key choices that need to be made during the design of steganographic benchmark, the first three seem to be infeasible. It is hard to imagine that in near future it would be possible to work in the space of all covers $\mathcal{C}$, or that a very good analytical model of natural images would be derived. Consequently, steganographic security can be verified only with respect to selected feature set and image database.

The issue tackled in this chapter is the possibility to replace tha last two choices (the classifier and the selection of a point on the ROC curve) with an other quantity free from any arbitrary choices. Such a benchmark would be more versatile and easier to use.

8.1. KL-Divergence

Since steganographic security can be in practice verified only with respect to steganographic features, it makes sense to appropriately change Cachin's definition:

Definition 8.1.1. Steganographic scheme (algorithm) is *secure with respect to feature set* $\mathbf{f} : \mathcal{C} \mapsto \mathcal{X} = \mathbb{R}^d$, if the Kullback–Leibler divergence between the probability distributions of cover, p_c, and stego objects, p_s, on the model space $\mathcal{X}$ is equal to zero:

$$D(p_c||p_s) = \sum_{\mathbf{x}\in\mathcal{X}} p_c(\mathbf{x}) \log \frac{p_c(\mathbf{x})}{p_s(\mathbf{x})} = 0. \tag{8.1.1}$$

When $D(p_c||p_s) < \epsilon$, the stego scheme is called ϵ-secure with respect to $\mathbf{f}$.

Definition 8.1.1 suggests that the preferable quantity for benchmarking steganographic schemes would be the KL divergence. KL divergence is justified by the detection theory, since it provides an upper bound on the best detector Eve can construct. The question is, if it is possible to estimate (8.1.1) precisely enough under conditions expected in steganography. Depending on the source of the cover images, for most scenarios it seems reasonable to assume that the sets

$$\begin{aligned} \mathbf{X} &= \left\{\mathbf{x}_i \in \mathbb{R}^d | \mathbf{x}_i \sim p_c, i \in \{1, \dots l\}\right\}, \\ \mathbf{Y} &= \left\{\mathbf{y}_i \in \mathbb{R}^d | \mathbf{y}_i \sim p_s, i \in \{1, \dots l\}\right\} \end{aligned} \tag{8.1.2}$$

were created from $l \approx 10^3 - 10^5$ images. Although in some cases [**36**] benchmark was performed on larger number of images, if the benchmark should be practical, the assumption made above on the number of images seems to be right, as it does not incur impractical computing requirements and storage. Most feature sets aspiring to be complete have dimension $d \approx 10 - 10^3$. The large dimensionality of $\mathcal{X}$ eliminates most estimators of KL divergence that can be potentially used. The only estimator that can provide accurate results in high dimensional spaces is the kNN estimator [**7, 62, 70**], which is briefly described in the next section. A good overview of the related problem of entropy estimation is provided in [**6**].

[1] $H(x)$ is the binary entropy function.

8.1.1. The kNN estimator of KL divergence. The kNN estimator of the KL divergence [**70**] is based on the kNN estimator of probability density. Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\} \in \mathcal{X}^l$, $\mathcal{X} = \mathbb{R}^d$ be a set of iid samples $\mathbf{x}_i \sim p_c$. By denoting $V_k(\mathbf{x}_i, \mathbf{X})$ the volume of a ball in $\mathcal{X}$ centered at $\mathbf{x}_i$ containing exactly k nearest neighbors of $\mathbf{x}_i$, the estimate of the pdf p_c at $\mathbf{x}_i$ can be expressed as

$$\hat{p}_c(\mathbf{x}_i) = \frac{\frac{k}{l-1}}{V_k(\mathbf{x}_i, \mathbf{X})}. \tag{8.1.3}$$

For the volume $V_k(\mathbf{x}_i, \mathbf{X})$ holds

$$V_k(\mathbf{x}_i, \mathbf{X}) = \frac{\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}+1\right)} \rho_k(\mathbf{x}_i, \mathbf{X}),$$

where $\rho_k(\mathbf{x}_i, \mathbf{X})$ is the distance of k-th closest neighbor of $\mathbf{x}_i$ from the set $\mathbf{X} \backslash \{\mathbf{x}_i\}$, and Γ is the gamma function[2].

By expressing the KL divergence as the expected value of likelihood

$$D_{\mathrm{KL}}(p_c \| p_s) = \int\limits_{\mathbb{R}^d} p_c(\mathbf{x}) \log \frac{p_c(\mathbf{x})}{p_s(\mathbf{x})} \mathrm{d}\mathbf{x} = \mathrm{E}_{\mathbf{x} \sim p_c} \left[\log \frac{p_c(\mathbf{x})}{p_s(\mathbf{x})} \right], \tag{8.1.4}$$

the kNN estimator of KL divergence is obtained by substituting estimate (8.1.3) for pdfs $p_c(\mathbf{x})$ and $p_s(\mathbf{x})$, and replacing the expectation $\mathrm{E}_{\mathbf{x} \sim p_c}$ by the average over all samples from $\mathbf{X}$. The formula for the kNN estimator is

$$\hat{D}_{\mathrm{KL}}(\mathbf{X}, \mathbf{Y}) = \frac{1}{l} \sum_{i=1}^{l} \log \frac{\hat{p}_c(\mathbf{x}_i)}{\hat{p}_s(\mathbf{x}_i)} = \frac{d}{l} \sum_{i=1}^{l} \log \frac{\rho_k(\mathbf{x}_i, \mathbf{Y})}{\rho_k(\mathbf{x}_i, \mathbf{X})} - \log \frac{l}{l-1}. \tag{8.1.5}$$

In [**70**] is showed that (8.1.5) is a consistent and asymptotically unbiased estimator of the KL divergence as long as $k/l \to 0$, and $k \to \infty$ as $l \to \infty$. For large l, the last term in (8.1.5) is approximately zero. Moreover, [**48**] showed that $\hat{D}_{\mathrm{KL}}(\mathbf{X}, \mathbf{Y}) \xmapsto[l \to \infty]{\text{a.s.}} D_{\mathrm{KL}}(p_c \| p_s)$ and that $\hat{D}_{\mathrm{KL}}(\mathbf{X}, \mathbf{Y})$ is unbiased if $p_c = p_s$. The parameter k (the number of nearest neighbors) controls the trade-off between bias and variance of the estimator. With increasing k, the bias of $\hat{D}_{\mathrm{KL}}(\mathbf{X}, \mathbf{Y})$ increases, but its variance decreases.

8.1.2. Tests on artificial data set. The accuracy of the kNN estimator under conditions assumed in steganalysis was verified on two experiments with artificially generated data. The first experiment investigates, how the kNN estimator scales with the dimension d, while the second one examines, how the bias attenuates as both probability distributions are coming closer to each other ($D_{\mathrm{KL}}(p_c \| p_s) \to 0$).

8.1.2.1. *Multivariate Gaussian.* In this experiment, synthetic data as generated from two d dimensional multivariate Gaussian distributions $p_c = N(-\mu, \mathbf{I})$ and $p_s = N(\mu, \mathbf{I})$, where $\mathbf{I}$ is the identity matrix and $\mu = \frac{1}{\sqrt{d}} \cdot \mathbf{1}$ with $\mathbf{1}$ being the vector of d ones. Notice that the mean of both Gaussians changes in such a way that the KL-divergence is constant ($D_{\mathrm{KL}}(p_c \| p_s) = 2$) through all dimensions. The accuracy of the kNN estimator was estimated on the following combinations of $2 \times l \in \{500, 1000, 5000, 10000, 50000, 100000\}$ samples and $d \in \{1, 5, 10, 100, 200, 300\}$ dimensions. In order to achieve the lowest bias of the estimator, k was set to 1. Since when $k = 1$, the variance of the estimator is high, each experiment was repeated 100 times. Table 8.1 shows the relative error $\frac{D_{\mathrm{KL}}(p,q) - \hat{D}_{\mathrm{KL}}(\mathbf{X}, \mathbf{Y})}{D_{\mathrm{KL}}(p,q)}$ of averages of estimates from 100 repetitions.

[2] $\Gamma(z) = \int_0^\infty t^{z-1} \exp(-t) \mathrm{d}t$.

d	2×500	2×1000	2×5000	2×10000	2×50000	2×100000
1	3.73%	2.86%	1.28%	1.45%	0.63%	0.15%
5	22.40%	18.73%	13.85%	11.70%	7.74%	6.64%
10	30.31%	27.01%	23.49%	21.55%	17.94%	16.23%
100	37.97%	43.20%	41.07%	42.11%	40.58%	39.34%
200	48.23%	42.35%	44.66%	44.56%	43.86%	42.40%
300	49.24%	48.64%	44.91%	44.54%	42.69%	48.85%

TABLE 8.1. Relative error of the KL-divergence estimate for two multi-variate Gaussian distributions for various combinations of sample sizes, l, and data dimensionality d. The number of nearest neighbors was set to $k = 1$ in order to achieve minimal bias of the estimator.

d	$\mu = \frac{1}{\sqrt{d}}$	$\mu = \frac{1}{2 \cdot \sqrt{d}}$	$\mu = \frac{1}{4 \cdot \sqrt{d}}$	$\mu = \frac{1}{8 \cdot \sqrt{d}}$	$\mu = \frac{1}{16 \cdot \sqrt{d}}$	$\mu = \frac{1}{32 \cdot \sqrt{d}}$	$\mu = 0$
1	0.23%	49.98%	75.17%	87.19%	93.71%	98.00%	$1.250348e - 03$
10	16.60%	57.04%	78.40%	88.55%	92.88%	95.51%	$2.804517e - 03$
100	40.43%	70.02%	83.78%	91.28%	98.97%	93.33%	$4.165952e - 03$

TABLE 8.2. Relative error of kNN estimators of KL-divergence estimate on multi-variate Gaussian distributions with data dimensionality $d \in \{1, 10, 100\}$ and converging means. The last column denoted $\mu = 0$ shows an absolute value of estimates for the case, when both distributions are the same ($D_{\mathrm{KL}}(p_c, p_s) = 0$). Estimates were calculated from 2×10^5 samples. The number of nearest neighbors was set to $k = 1$ in order to achieve minimal bias of the estimator.

It is obvious that the estimates are clearly biased and this bias tends to zero very slowly with increasing number of data samples (it has to go to zero because the estimator (8.1.5) is asymptotically unbiased). The high bias comes from the estimation of cross-entropy. While entropy can be estimated accurately even in high-dimensional spaces with small number of data samples, the cross-entropy is harder to estimate. This is because $\log q(\mathbf{x})$ needs to be estimated at regions where $p(\mathbf{x})$ is large, but $q(\mathbf{x})$ is different, there may not be enough data points from $\mathbf{Y}$ to estimate $\log q(\mathbf{x})$ accurately. This problem of slow convergence of the cross-entropy estimator persists for other distributions, such as the Student's t-distribution, which seems to be a relevant model of output from some LSB detectors [**35**].

8.1.2.2. *Convergence speed with respect to the distance between means.* The previous experiment shows that the bias of kNN estimator of the KL divergence comes from the estimation of cross-entropy. In the verification of steganographic schemes, the distributions p_c and p_s are close to each other. Therefore, it can be expected that the accuracy of the kNN estimator will improve, as the distributions become more similar.

In this experiment, the number of samples l was kept same at $l = 2 \times 10^5$. Similarly to the previous section, data was generated according to two d dimensional multivariate Gaussian distributions $p_c = N(-\mu, \mathbf{I})$ and $p_s = N(\mu, \mathbf{I})$ with

$$\mu \in \left\{ \frac{1}{\sqrt{d}2^i} \cdot \mathbf{1} | i \in \{0, 1, 2, 3, 4, 5\}, d \in \{1, 10, 100\} \right\} \cup \{\mathbf{0}\}.$$

The corresponding KL divergences are $\left(2, \frac{1}{2}, \frac{1}{8}, \frac{1}{32}, \frac{1}{128}, \frac{1}{512}, 0\right)$. Again, $k = 1$.

Table 8.2 shows relative errors from average of 100 runs with $k = 1$ (absolute values of KL divergence are shown for $\mu = \mathbf{0}$ case). Contrary to the expectation, the relative error of the estimates increases with decreasing μ. This shows that the bias of the estimator is relatively large even for cases when the pdfs are quite similar. Consequently, the kNN estimator cannot be used to verify the security of steganographic schemes.

8.1.3. KL-divergence in steganography. Without any doubts, the KL divergence in the model space $\mathcal{X}$ is the preferable quantity for benchmarking steganographic schemes, because it provides fundamental information about the limits of any steganalytic method. Moreover, it could be used for evaluating the suitability of different feature sets to distinguish between cover and stego objects for a fixed steganographic method (obtaining thus an interesting steganalysis benchmark). The presented results on artificial problems with conditions similar to steganography (dimension $d \sim 10 - 300$ and $l \lesssim 10^5$ samples) show, that existing estimators do not possess sufficient accuracy. The effort to remedy this situation could be directed towards deriving better behaved bias-free estimators of KL divergence and reducing the dimensionality of the model space [**43**].

8.2. Maximum Mean Discrepancy (MMD)

The results presented in previous section exhibit the weakness of the KL Divergence — it cannot be reliably estimated on spaces with higher dimension, unless some side knowledge is utilized. Therefore, if one wants to verify the security of a steganographic algorithm, an alternative method needs to be used. From the definition of steganographic security 8.1.1 it follows that the steganographic scheme $(S_{\mathrm{E}}, S_{\mathrm{X}})$ is secure with respect to feature set $\mathbf{f}$ iff $D_{\mathrm{KL}}(p_c \| p_s) = 0$. Even though the KL divergence is not a true distance (it is not symmetric) measure, it possesses an important property of a distance, namely that

$$(D_{\mathrm{KL}}(p_c \| p_s) = 0) \Longleftrightarrow (p_c = p_s)\,.$$

This property allows to formulate a practical test of steganographic security as a two-sample hypothesis test [**24**]

$$\begin{aligned} \mathrm{H}_0 : &\quad p_c = p_s \\ \mathrm{H}_1 : &\quad p_c \neq p_s \quad . \end{aligned} \tag{8.2.1}$$

The goal of the two-sample hypothesis test 8.2.1 is to verify, if probability distributions of cover, p_c, and stego objects, p_s, are the same. If so ($p_c = p_s$, hypothesis H_0 is true), then the steganographic scheme is secure (with respect to feature set $\mathbf{f}$). From among available methods for the two-sample problem (see the review in, e.g., [**24**]), the Maximum Mean Discrepancy (MMD) [**24, 25**], seems to be the most suitable for steganography due to following advantages:

- MMD is numerically stable and scales very well with data dimensionality. It has been shown that MMD converges almost independently of data dimension d with error $1/\sqrt{l}$, where l is the number of samples. Fast convergence provides an accurate benchmark from even $\sim 10^3$ images. Some experimental results on artificial data sets showing this phenomenon are presented in Section 8.3.
- MMD has strong theoretical foundations and can be linked to other methods, such as Parzen Window estimates.
- MMD's computational complexity is $O(l^2)$, which is fast in comparison to Support Vector Machines (SVM) (complexity of training is $O(l^3)$ + grid-search for hyper parameters).

In the rest of this section, MMD is presented in more detail. To this end, it is assumed that $\mathcal{X}$ is a separable metric space (this holds in steganography, as $\mathcal{X} = \mathbb{R}^d$), and p_c, p_s are probability distributions defined on $\mathcal{X}$. The main idea behind MMD is based on following the Lemma (9.3.2 of [**13**]):

Lemma 8.2.1. *$p_s = p_q$ if and only if $(\forall f \in \mathcal{C}(\mathcal{X}))(\mathbf{E}_{\mathsf{x}\sim p_c} f(\mathsf{x}) = \mathbf{E}_{\mathsf{y}\sim p_s} f(\mathsf{y}))$, where $\mathcal{C}(\mathcal{X})$ is the class of continuous bounded functions on $\mathcal{X}$.*

Lemma 8.2.1 cannot be directly used in practice, because the class of all continuous bounded functions on $\mathcal{X}$ is too rich to evaluate the condition of the lemma from finite number of samples. The avenue MMD takes to make Lemma 8.2.1 computationally tractable is to restrict the set of functions to a narrower class $\mathcal{F}$, and measure the disparity between p and q with respect to $\mathcal{F}$ as

$$\text{MMD}[\mathcal{F}, p, q] = \sup_{f \in \mathcal{F}} \left(\mathbf{E}_{\mathsf{x}\sim p_c} f(\mathsf{x}) - \mathbf{E}_{\mathsf{y}\sim p_s} f(\mathsf{y})\right). \tag{8.2.2}$$

In finite sample setting, (8.2.2) yields to

$$\text{MMD}[\mathcal{F}, \mathbf{X}, \mathbf{Y}] = \sup_{f \in \mathcal{F}} \left(\frac{1}{l}\sum_{i=1}^{l} f(x_i) - \frac{1}{l}\sum_{i=1}^{l} f(y_i)\right), \tag{8.2.3}$$

where $\mathbf{X} = \{x_1, \dots, x_l\}$, $\mathbf{Y} = \{y_1, \dots, y_l\}$ are samples (8.1.2) from p_c and p_s, respectively. The key ingredient of efficiency of MMD is the choice of class of functions $\mathcal{F}$. $\mathcal{F}$ needs to be rich to distinguish $p \neq q$, yet restrictive enough to provide useful finite sample estimates. MMD chooses $\mathcal{F}$ to be a unit ball in universal Reproducing Kernel Hilbert Space, which is described in the next section.

8.2.1. Reproducing Kernel Hilbert Spaces.

This subsection briefly introduces Reproducing Kernel Hilbert Spaces $\mathcal{H}$ (RKHS) by showing the construction of $\mathcal{H}$ from a symmetric positive definite function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ called *kernel* (each RKHS $\mathcal{H}$ is tightly linked to its kernel k). Kernel function is for a example Gaussian kernel

$$k(x, x') = \exp\left(-\gamma \|x - x'\|^2\right).$$

During the construction of $\mathcal{H}$, several interesting properties of RKHS are revealed. The introduction starts with recapitulation of the definition of a positive definite function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$.

Definition 8.2.2. Function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is positive definite iff

$$(\forall n \in \mathbb{N}_0)\,(\forall (x_1, \dots, x_n) \in \mathcal{X}^n)\,(\forall (c_1, \dots, c_n) \in \mathbb{R}^n)\left(\sum_{i,j=1}^{n,n} c_i c_j k(x_i, x_j) \geq 0\right).$$

The following property of positive definite kernel functions is very useful in further development of the RKHS.

Proposition 8.2.3. *If $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is positive definite kernel and $x_1, x_2 \in \mathcal{X}$, then*

$$k(x_1, x_2)^2 \leq k(x_1, x_1) k(x_2, x_2). \tag{8.2.4}$$

Proof. Since k is positive definite, the Gram matrix $\mathbf{K} \in \mathbb{R}^{2,2}$, $\mathbf{K}_{ij} = k(x_i, x_j)$ is positive definite as well. For the determinant of $\mathbf{K}$ holds

$$0 \leq |\mathbf{K}| = \mathbf{K}_{11}\mathbf{K}_{22} - \mathbf{K}_{12}\mathbf{K}_{21} = k(x_1, x_1)k(x_2, x_2) - k(x_1, x_2)^2,$$

from where the proposition follows. By using kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, every $x \in \mathcal{X}$ can be associated with function $K_x : \mathcal{X} \mapsto \mathbb{R}$ as $K_x(\cdot) = k(x, \cdot)$. The next definition introduces pre-Hilbert space $\mathcal{H}_0$[3] of all finite linear combinations of functions K_x, $x \in \mathcal{X}$. □

[3]Pre-Hilbert space is a linear space endowed with dot-product.

DEFINITION 8.2.4. The set $\mathcal{H}_0$ of all finite linear combinations of functions K_x, $x \in \mathcal{X}$

$$\mathcal{H}_0 = \left\{ \sum_{i=1}^{n} a_i K_{x_i} | n \in \mathbb{N}_0,\ a_i \in \mathbb{R},\ x_i \in \mathcal{X} \right\}$$

forms a linear vector space of functions $\mathcal{X} \mapsto \mathbb{R}$. A dot product on $\mathcal{H}_0$ between $f = \sum_{i=1}^{n} a_i K_{x_i}$ and $g = \sum_{j=1}^{m} b_j K_{x_j}$ can be defined as

$$\langle f, g \rangle_{\mathcal{H}_0} = \sum_{i=1}^{n} \sum_{j=1}^{m} a_i b_j k(x_i, y_j). \tag{8.2.5}$$

The definition of the dot-product 8.2.5 might seem dubious, as it contains expansion coefficients a_i and b_j. The next paragraph proves that the dot-product is indeed a well defined dot-product.

The property

$$\begin{aligned} \langle f, g \rangle_{\mathcal{H}_0} = \quad \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} a_i b_j k(x_i, y_j) \quad &= \sum_{i=1}^{n} a_i \sum_{j=1}^{m} b_j k_{y_j}(x_i) = \sum_{i=1}^{n} a_i g(x_i) \\ &= \sum_{j=1}^{m} b_j \sum_{i=1}^{n} a_i k_{x_i}(y_j) = \sum_{j=1}^{m} b_j f(y_j) \end{aligned}$$

shows forms free from expansion coefficients either $(a_i)_{i=1}^{n}$ or $(b_j)_{j=1}^{m}$. Consequently, the value of the dot-product 8.2.5 does not depend on expansion coefficients. The very same property also shows that the dot-product is bi-linear. The symmetry and positive definiteness of the dot-product are guaranteed by the same features of the kernel function k. The dot-product itself is again a kernel function $\langle \cdot, \cdot \rangle_{\mathcal{H}_0} : \mathcal{H}_0 \times \mathcal{H}_0 \mapsto \mathbb{R}$, since it holds

$$(\forall n \in \mathbb{N}_0)\,(\forall (x_1, \ldots, x_n) \in \mathcal{X}^n)\,(\forall (c_1, \ldots, c_n) \in \mathbb{R}^n)$$
$$\left(\sum_{i,j=1}^{n,n} c_i c_j \langle f_i, f_j \rangle_{\mathcal{H}_0} = \left\langle \sum_{i=1}^{n} c_i f_i, \sum_{j=1}^{n} c_j f_j \right\rangle_{\mathcal{H}_0} \geq 0 \right).$$

Therefore the value of the dot product is always non-negative. The last condition that needs to be verified is that $f(x) = 0$ iff $\|f\|_{\mathcal{H}_0} = 0$. From the definition of the dot-product (8.2.5) it follows that $(\forall f = \sum_{i=1}^{n} \alpha_i K_{x_i} \in \mathcal{H}_0)\,(\forall x \in \mathcal{X})$

$$\langle f, K_x \rangle_{\mathcal{H}_0} = \left\langle \sum_{i=1}^{n} \alpha_i K_{x_i}, K_x \right\rangle_{\mathcal{H}_0} = \sum_{i=1}^{n} \alpha_i \langle K_{x_i}, K_x \rangle_{\mathcal{H}_0} = \sum_{i=1}^{n} \alpha_i k(x_i, x) = f(x). \tag{8.2.6}$$

Finally,

$$(\forall f \in \mathcal{H}_0)\,(x \in \mathcal{X})\, |f(x)|^2 = \langle f, k(x, \cdot) \rangle^2 \leq k(x, x) \cdot \|f\|_{\mathcal{H}_0} \tag{8.2.7}$$

shows that $f(x) = 0$ iff $\|f\|_{\mathcal{H}_0} = 0$, which completes the proof that (8.2.5) is a dot-product.

The construction of the Hilbert space[4] $\mathcal{H}$ is finished by completing pre-Hilbert $\mathcal{H}_0$ space with respect to the norm $\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}}$.

Hilbert space $\mathcal{H}$ contains functions $f : \mathcal{X} \mapsto \mathbb{R}$ that can be arbitrarily precisely approximated by finite linear combinations of $K_x = k(x, \cdot)$ centered at finite number of points x. Properties (8.2.6) and (8.2.7) can be extended to all functions $f \in \mathcal{H}$. Property (8.2.6) is called *reproducing property*. It is frequently formulated as follows: *For each $x \in \mathcal{X}$, the point evaluation functional $\delta_x : \mathcal{H} \mapsto \mathbb{R}$, $\delta_x(f) = f(x)$, is a continuous linear functional.*

[4]Hilbert space $\mathcal{H}$ is a vector space with dot-product $\langle f, g \rangle_{\mathcal{H}}$ complete with respect to the norm defined as $\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}}$.

An important consequence of the property (8.2.7) is that convergence in norm $\|f\|_{\mathcal{H}}$ implies point-wise convergence.

Hilbert spaces constructed by means of kernel k, as was shown above, are called *Reproducing Kernel Hilbert Spaces* (RKHS). RKHS is uniquely defined by its kernel (kernel defining RKHS is called reproducing) and vice versa, RKHS has only one reproducing kernel. Because of this equivalence, RKHS can be alternatively defined as follows (see for example [**68**]).

Definition 8.2.5. Let $\mathcal{X} \neq 0$ and $\mathcal{H}$ be a Hilbert space of functions $\mathcal{X} \mapsto \mathbb{R}$. The space $\mathcal{H}$ is called RKHS over $\mathcal{X}$ iff $\forall x \in \mathcal{X}$ point evaluation functionals $\delta_x : \mathcal{X} \mapsto \mathbb{R}$, $\delta_x(f) = f(x)$ are linear and continuous. A function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is called a reproducing kernel of $\mathcal{H}$ iff $(\forall x \in \mathcal{X})$ $(K_x = k(x, \cdot) \in \mathcal{H})$ and $(f(x) = \langle f, k(x, \cdot)\rangle_{\mathcal{H}})$.

In the machine-learning literature dealing with kernels, the reader can commonly encounter the following definition linking *kernel* function to the *feature space* (see for example [**68**], Appendix A.2).

Definition 8.2.6. Let $\mathcal{X}$ be a non-empty set. Then function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is called kernel on $\mathcal{X}$ iff there exists space $\mathcal{H}^{'}$ and mapping $\phi : \mathcal{X} \mapsto \mathcal{H}^{'}$ such that

$$\left(\forall x, x^{'} \in \mathcal{X}\right) \left(k(x, x^{'}) = \left\langle \phi(x), \phi(x^{'}) \right\rangle_{\mathcal{H}^{'}}\right).$$

Mapping ϕ is called the *feature map* and $\mathcal{H}^{'}$ is called *feature space* of k.

The reason, why this definition is brought up here is to make clear the difference between feature space and Reproducing Kernel Hilbert Space. Feature space $\mathcal{H}^{'}$ does not necessarily possess the reproducing property and therefore it is not unique to the kernel. For a single kernel k, more than one feature map ϕ with feature space $\mathcal{H}^{'}$ can exist, but kernel k has only one RKHS $\mathcal{H}$. For a concrete example, see [**68**].

As can be seen from above, RKHS is tightly linked to its kernel. An important class of *universal* kernels is identified in [**65**].

Definition 8.2.7. Kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is universal iff $\mathcal{X}$ is compact and its RKHS is dense in $C(\mathcal{X})$ in the maximum (infinity) norm $\|f - g\|_{\infty} = \sup_{x \in \mathcal{X}} |f(x) - g(x)|$.

Universal kernels play a fundamental role in the theory behind Support Vector Machines. In [**65**] is showed that Support Vector Machines equipped with universal kernel converge to Bayes-optimal classifier. A universal kernel, which is exclusively used in this dissertation, is the Gaussian kernel on $\mathcal{X} \subset \mathbb{R}^d$

$$k(x, y) = \exp(-\gamma \|x - y\|_2^2),\ \gamma > 0. \tag{8.2.8}$$

For examples of other universal kernels, see [**65**].

8.2.2. MMD. The next theorem due to [**24**] shows why the RKHS corresponding to the universal kernel is a good choice in MMD.

Theorem 8.2.8. *Let $\mathcal{F}$ be a unit ball in a universal RKHS $\mathcal{H}$. Then* $\mathrm{MMD}[\mathcal{F}, p_c, p_s] = 0$ *if and only if $p_c = p_s$.*

Proof. $\Leftarrow$: Follows immediately from the definition of MMD (8.2.2).

$\Rightarrow$: The reverse implication is a simple consequence of Lemma 8.2.1 and denseness of universal RKHS in $\mathcal{C}(\mathcal{X})$. It is proved by showing that if $\mathrm{MMD}[C(\mathcal{X}), p_c, p_s] = D > 0$ then $\mathrm{MMD}[\mathcal{F}, p_c, p_s] > 0$. Since $p_s \neq p_c$ implies $\mathrm{MMD}[C(\mathcal{X}), p_c, p_s] > 0$, the implication is proved from Lemma 8.2.1.

Let us assume that $\mathrm{MMD}[C(\mathcal{X}), p_s, p_c] = D > 0$. Then there exist $\tilde{f} \in C(\mathcal{X})$ such that

$$\mathbf{E}_{\mathrm{x}\sim p_c}[\tilde{f}(\mathrm{x})] - \mathbf{E}_{\mathrm{y}\sim p_s}[\tilde{f}(\mathrm{y})] \geq \frac{D}{2}. \tag{8.2.9}$$

Because universal RKHS $\mathcal{H}$ is dense in $C(\mathcal{X})$ with respect to the L_∞ norm, there has to exist $f^* \in \mathcal{H}$ such that $\|\tilde{f} - f^*\|_\infty < \epsilon < \frac{D}{8}$. Since the expectation $\mathbf{E}_{\mathrm{x}}$ is continuous, for the difference between expectations $\mathbf{E}_{\mathrm{x}}[\tilde{f}(\mathrm{x})]$ and $\mathbf{E}_{\mathrm{x}}[f^*(\mathrm{x})]$ holds

$$\left|\mathbf{E}_{\mathrm{x}}[\tilde{f}(\mathrm{x})] - \mathbf{E}_{\mathrm{x}}[f^*(\mathrm{x})]\right| < \epsilon < \frac{D}{8}. \tag{8.2.10}$$

By using inequalities (8.2.9) and (8.2.10), following lower bound on the difference between expectations $\mathbf{E}_{\mathrm{x}\sim p_c}[f^*(\mathrm{x})]$ and $\mathbf{E}_{\mathrm{x}\sim p_s}[f^*(\mathrm{x})]$ is derived:

$$\begin{aligned}
\left|\mathbf{E}_{\mathrm{x}\sim p_c}[f^*(\mathrm{x})] - \mathbf{E}_{\mathrm{y}\sim p_s}[f^*(\mathrm{y})]\right| &= \Big|\mathbf{E}_{\mathrm{x}\sim p_c}[f^*(\mathrm{x}) - \tilde{f}(\mathrm{x}) + \tilde{f}(\mathrm{x})] \\
&\quad -\mathbf{E}_{\mathrm{y}\sim p_s}[f^*(\mathrm{y}) - \tilde{f}(\mathrm{y}) + \tilde{f}(\mathrm{y})]\Big| \\
&= \Big|\Big[\mathbf{E}_{\mathrm{x}\sim p_c}[f^*(\mathrm{x}) - \tilde{f}(\mathrm{x})] - \mathbf{E}_{\mathrm{y}\sim p_s}[f^*(\mathrm{y}) - \tilde{f}(\mathrm{y})]\Big] \\
&\quad + \Big[\mathbf{E}_{\mathrm{x}\sim p_c}[\tilde{f}(\mathrm{x})] - \mathbf{E}_{\mathrm{y}\sim p_s}[\tilde{f}(\mathrm{y})]\Big]\Big| \\
&> \left|\mathbf{E}_{\mathrm{x}\sim p_c}[\tilde{f}(\mathrm{x})] - \mathbf{E}_{\mathrm{y}\sim p_s}[\tilde{f}(\mathrm{y})]\right| \\
&\quad - \left|\mathbf{E}_{\mathrm{x}\sim p_c}[f^*(\mathrm{x}) - \tilde{f}(\mathrm{x})] - \mathbf{E}_{\mathrm{y}\sim p_s}[f^*(\mathrm{y}) - \tilde{f}(\mathrm{y})]\right| > \\
&\underset{(8.2.10)}{>} \left|\mathbf{E}_{\mathrm{x}\sim p_c}[\tilde{f}(\mathrm{x})] - \mathbf{E}_{\mathrm{y}\sim p_s}[\tilde{f}(\mathrm{y})]\right| - 2\frac{D}{8} \\
&\underset{(8.2.9)}{\geq} \frac{D}{2} - 2\frac{D}{8} = \frac{D}{4}.
\end{aligned}$$

The proof is finished by rescaling f^* so that $\frac{f^*}{\|f^*\|_{\mathcal{H}}} \in \mathcal{F}$ ($\mathcal{F}$ is a unit ball). □

If the kernel corresponding to RKHS is bounded ($\sup_{x\in\mathcal{X}} k(x,x) = \kappa < +\infty$), $\mathrm{MMD}[\mathcal{F}, p_c, p_s]$ defined in (8.2.2) accepts a particularly simple form. Before the analytical form is shown, it is useful to introduce the mean functions $\mu_{p_c}, \mu_{p_s} \in \mathcal{H}$ with the following property

$$(\forall f \in \mathcal{H})\left(\mathbf{E}_{\mathsf{x}\sim p_c} f(\mathsf{x}) = \langle f, \mu_{p_c}\rangle_{\mathcal{H}} \text{ and } \mathbf{E}_{\mathsf{x}\sim p_s} f(\mathsf{x}) = \langle f, \mu_{p_s}\rangle_{\mathcal{H}}\right).$$

The existence of μ_{p_c} and μ_{p_s} follows from the Riesz theorem (expectation operator $\mathbf{E}$ is linear and in RKHS with bounded kernel it is bounded[5]). The equality

$$\mathbf{E}_{\mathsf{x}\sim p_c} f(\mathsf{x}) = \mathbf{E}_{\mathsf{x}\sim p_c}\langle f, k(\mathsf{x},\cdot)\rangle_{\mathcal{H}} = \langle f, \mathbf{E}_{\mathsf{x}\sim p_c}[k(\mathsf{x},\cdot)]\rangle_{\mathcal{H}} = \langle f, \mu_{p_c}\rangle_{\mathcal{H}} \tag{8.2.11}$$

shows that $\mu_{p_c} = \mathbf{E}_{\mathsf{x}\sim p_c}[k(\mathsf{x},\cdot)]$. The same holds for μ_{p_s}, as the derivations do not depend on the distribution.

The analytical form of $\mathrm{MMD}[\mathcal{F}, p_c, p_s]$ can be obtained by using equality (8.2.11) to (8.2.2) as follows

$$\mathrm{MMD}[\mathcal{F}, p_c, p_s] = \sup_{f\in\mathcal{F}}\left(\mathbf{E}_{\mathsf{x}\sim p_c} f(\mathsf{x}) - \mathbf{E}_{\mathsf{y}\sim p_s} f(\mathsf{y})\right) = \sup_{\|f\|_{\mathcal{H}}\leq 1}\langle f, \mu_{p_c} - \mu_{p_s}\rangle = \|\mu_{p_c} - \mu_{p_s}\|_{\mathcal{H}}. \tag{8.2.12}$$

The last equality is a simple consequence of the conditions for equality in Cauchy-Schwartz inequality, namely that the supremum is reached for $f = (\mu_p - \mu_q)/\|\mu_p - \mu_q\|_{\mathcal{H}}$.

[5]Because $f \in \mathcal{F}$, $\|f\|_{\mathcal{H}} \leq 1$. Therefore $\int f(x)p(x)\mathrm{d}x = \int_{\mathcal{X}}\langle f, k(x,\cdot)\rangle_{\mathcal{H}}\, p(x)\mathrm{d}x \underset{(8.2.7)}{\leq} \int_{\mathcal{X}} |k(x,x)|\|f\|_{\mathcal{H}} p(x)\mathrm{d}x \leq k\int_{\mathcal{X}} p(x)\mathrm{d}x = k$.

8.2.2.1. *Biased empirical estimator.* The empirical estimators of MMD are derived from its analytical form.

Assuming set of samples (8.1.2), a simple estimator of MMD

$$\mathrm{MMD}_b[\mathcal{F},\mathbf{X},\mathbf{Y}] = \left[\frac{1}{l^2}\sum_{i,j=1}^{l,l} k(x_i,x_j)+k(y_i,y_j)-2k(x_i,y_j)\right]^{\frac{1}{2}} \tag{8.2.13}$$

is obtained by replacing means μ_{p_c} and μ_{p_s} in (8.2.12) by corresponding estimates $\hat{\mu}_{p_c}(x) = \frac{1}{l}\sum_{i=1}^{l} k(x_i,x)$ and $\hat{\mu}_{p_s}(x) = \frac{1}{l}\sum_{i=1}^{l} k(y_i,x)$. Denoting

$$h(x,x^{'},y,y^{'}) = k(x,x^{'})+k(y,y^{'})-k(x,y^{'})-k(x^{'},y), \tag{8.2.14}$$

estimator (8.2.13) can be compactly written as

$$\mathrm{MMD}_b[\mathcal{F},\mathbf{X},\mathbf{Y}] = \left[\frac{1}{l^2}\sum_{i,j=1}^{l,l} h(x_i,x_j,y_i,y_j)\right]^{\frac{1}{2}}.$$

In the notation of U-statistics (see [**60**]), the function $h(x_i,x_j,y_i,y_j)$ is called *kernel* (do not mistake with kernel function k used in the theory of RKHS). Despite the fact that estimator (8.2.13) is biased, its appealing property is that errors of estimates under both hypothesis can be bounded.

Theorem 8.2.9. *Let p_c, p_s, $\mathbf{X}$, $\mathbf{Y}$, and $\mathcal{F}$ be defined as above and reproducing kernel k is bounded by κ, such that $(\forall x,y\in\mathcal{X})$ $|k(x,y)|\leq\kappa$. Then,*

$$Pr\left\{|\mathrm{MMD}_b[\mathcal{F},\mathbf{X},\mathbf{Y}]-\mathrm{MMD}[\mathcal{F},p_c,p_s]| > 4\sqrt{\frac{\kappa}{l}}+\epsilon\right\} \leq 2\exp\left(\frac{-\epsilon^2 l}{4\kappa}\right).$$

Theorem 8.2.10. *Under the assumption of Theorem 8.2.9, where additionally $p_c = p_q$*

$$P\left\{\mathrm{MMD}_b[\mathcal{F},\mathbf{X},\mathbf{Y}] > \sqrt{\frac{2\kappa}{l}}+\epsilon\right\} \leq \exp\left(-\frac{\epsilon^2 l}{4\kappa}\right).$$

Both theorems 8.2.9 and 8.2.10 above (due to Gretton et al. [**25**]) show that the bias of the estimator attenuates with square root of the number of examples l. Therefore the two sample tests based on estimator MMD_b are asymptotically consistent. For Gaussian kernel, which is the kernel of choice for most application in steganography, it holds that $\kappa = 1$. Therefore, the bias of the estimator is less than $\sqrt{\frac{2\kappa}{l}}$. The following corollary is an immediate consequence of Theorem 8.2.10. It expresses the level of acceptance of hypothesis H_0 of two-sample test based on the biased estimate $\mathrm{MMD}_b(\mathcal{F},\mathbf{X},\mathbf{Y})$.

Corollary 8.2.11. *A hypothesis test of level α (probability of false rejection) for the hypothesis* H_0 *(equivalently* $\mathrm{MMD}[\mathcal{F},p_c,p_s]=0$*) has the acceptance region*

$$\mathrm{MMD}_b[\mathcal{F},\mathbf{X},\mathbf{Y}] < \sqrt{\frac{2\kappa}{l}}\left(1+\sqrt{-2\log\alpha}\right).$$

8.2.2.2. *Unbiased empirical estimator.* An unbiased estimator of $\mathrm{MMD}(\mathcal{F},p_c,p_s)$ can be derived from U-statistics, as is shown in [**24**]. An unbiased estimator is similar to the biased estimator MMD_b except that it skips samples with same indices in the evaluation of the kernel. By using function $h(x,x^{'},y,y^{'})$ (8.2.14), the unbiased estimator can be written as

$$\mathrm{MMD}_u^2(\mathcal{F}, \mathbf{X}, \mathbf{Y}) = \frac{1}{l(l-1)} \sum_{\substack{i,j=1 \\ i \neq j}}^{l} h(x_i, x_j, y_i, y_j). \tag{8.2.15}$$

THEOREM 8.2.12. *Let assume that* $\mathbf{E}_{\mathsf{x},\mathsf{x}' \sim p_c, \mathsf{y},\mathsf{y}' \sim p_s}\left[h(\mathsf{x},\mathsf{x}',\mathsf{y},\mathsf{y}')\right] < \infty$.

(1) *If* $p_c \neq p_s$*, then* MMD_u^2 *converges in distribution to a Gaussian according to*

$$\sqrt{l}\left(\mathrm{MMD}_{\mathrm{u}}^2[\mathcal{F}, \mathbf{X}, \mathbf{Y}] - \mathrm{MMD}^2[\mathcal{F}, p, q]\right) \xrightarrow{D} \mathcal{N}(0, \sigma_u^2),$$

where

$$\sigma_u^2 = 4\left(\mathbf{E}_{\mathsf{x}\sim p_c, \mathsf{y}\sim p_s}\left[\left(\mathbf{E}_{\mathsf{x}'\sim p_c, \mathsf{y}'\sim p_s} h(\mathsf{x},\mathsf{x}',\mathsf{y},\mathsf{y}')\right)^2\right] - \left[\mathbf{E}_{\mathsf{x},\mathsf{x}'\sim p_c, \mathsf{y},\mathsf{y}'\sim p_s}\left(h(\mathsf{x},\mathsf{x}',\mathsf{y},\mathsf{y}')\right)\right]^2\right).$$

(2) *If* $p_c = p_s$*, the U-statistic is degenerate, which means that mean* $\mathbf{E}_{\mathsf{x}'\sim p_c, \mathsf{y}'\sim p_s} h(\mathsf{x},\mathsf{x}',\mathsf{y},\mathsf{y}') = 0$*. In this case,* MMD_u^2 *converges according to*

$$l \cdot \mathrm{MMD}_u^2 \xrightarrow{D} \sum_{r=1}^{\infty} \lambda_r [z_r^2 - 2], \tag{8.2.16}$$

where $z_r \sim \mathcal{N}(0, 2)$ *iid, and* λ_r *are solutions of the eigenvalue equation*

$$\int_{\mathcal{X}} \tilde{k}(x, x') \psi_r(x) \mathrm{d}p(x) = \lambda_r \psi_r(x'),$$

with $\tilde{k}(x, x') = k(x, \mathsf{x}') - \mathbf{E}_{\mathsf{x}\sim p_c} k(x, \mathsf{x}) - \mathbf{E}_{\mathsf{x}\sim p_c} k(\mathsf{x}, x) + \mathbf{E}_{\mathsf{x},\mathsf{x}'\sim p_c} k(\mathsf{x}, \mathsf{x}')$ *being the centered RKHS kernel.*

The probability distribution (8.2.16) is generally impossible to calculate, since just the calculation of the eigenvalues λ_r can be very difficult even in cases, when the pdf p_c is known. Two methods to approximate the distribution of $\mathrm{MMD}_u[\mathcal{F}, \mathbf{X}, \mathbf{Y}]^2$ under H_0 are proposed in [**25**]. The first method fits the Pearson curve into first four moments of $\mathrm{MMD}_u[\mathcal{F}, \mathbf{X}, \mathbf{Y}]^2$, the second approach uses bootstrapping [**1**] on aggregated data. Since the experimental results of both methods were similar, the latter method is recommended especially for large data sets, since its computational cost is significantly lower.

8.2.2.3. *Analytical calculation of MMD.* This section shows, how to calculate analytically MMD, when Gaussian kernel $k : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$, $k(x, y) = \exp\left(-\gamma(x-y)^2\right)$ is used. This analytical evaluation is used in Section 8.3.1, where convergence rates of MMD on selected probability distributions are shown.

According to [**68**], an orthonormal basis of RKHS spanned by Gaussian kernel can be expressed as follows

$$\left\{ e_n(y) = \sqrt{\frac{(2\gamma)^n}{n!}} y^n \exp(-\gamma y^2) \,\middle|\, n \geq 0 \right\}.$$

Known ON basis enables us to evaluate the norm in (8.2.12) as

$$\mathrm{MMD}^2[\mathcal{F}, p_c, p_s] = \|\mu_{p_c} - \mu_{p_s}\|_{\mathcal{F}}^2 = \sum_{n=0}^{\infty} \left(b_n^{p_c} - b_n^{p_s}\right)^2,$$

where $b_n^{p_c} = \langle \mu_{p_c}, e_n \rangle_{\mathcal{H}}$ and similarly $b_n^{p_s} = \langle \mu_{p_s}, e_n \rangle_{\mathcal{H}}$. From (8.2.11), (8.2.6), and (8.2.5) follows that

$$\begin{aligned}\mu_{p_c}(y) &= \langle \mu_{p_c}, k(y,\cdot)\rangle_{\mathcal{H}} = \mathbf{E}_{\mathsf{x}\sim p_c}\langle k(\mathsf{x},\cdot), k(y,\cdot)\rangle_{\mathcal{H}} = \mathbf{E}_{\mathsf{x}\sim p_c} k(\mathsf{x},y) = \int_{\mathbb{R}} p_c(x)\cdot k(x,y)\mathrm{d}x \\ &= \int_{\mathbb{R}} p_c(x)\cdot \exp(-\gamma(x-y)^2)\mathrm{d}x = \sum_{n=0}^{\infty} b_n^{p_c}\sqrt{\frac{(2\gamma)^n}{n!}}y^n\exp(-\gamma y^2).\end{aligned}$$

Multiplying the whole equation by $\exp(\gamma y^2)$,

$$\int_{\mathbb{R}} p_c(x)\cdot\exp(-\gamma(x^2-2xy))\mathrm{d}x = \sum_{n=0}^{\infty} b_n^{p_c}\sqrt{\frac{(2\gamma)^n}{n!}}y^n,$$

and by using Taylor expansion of function $\int_{\mathbb{R}} p(x)\cdot\exp(-\gamma(x^2-2xy))\mathrm{d}x$ at $y=0$, it holds that

$$\begin{aligned}\sum_{n=0}^{\infty} b_n^{p_c}\sqrt{\frac{(2\gamma)^n}{n!}}y^n &= \sum_{n=0}^{\infty}\frac{1}{n!}\frac{\partial^n}{\partial y^n}\left[\int_{\mathbb{R}} p_c(x)\cdot\exp(-\gamma(x^2-2xy))\mathrm{d}x\right]\Bigg|_{y=0} y^n = \\ &= \sum_{n=0}^{\infty}\frac{1}{n!}\left[\int_{\mathbb{R}}(2\gamma)^n x^n p_c(x)\cdot\exp(-\gamma x^2)\mathrm{d}x\right]y^n,\end{aligned}$$

and thus

$$b_n^{p_c} = \int_{\mathbb{R}} p(x)\cdot\sqrt{\frac{(2\gamma)^n}{n!}}x^n\exp(-\gamma x^2)\mathrm{d}x = \int_{\mathbb{R}} p_c(x)\cdot e_n(x)\mathrm{d}x.$$

In other words, coefficients $b_n^{p_c}$ are equal to the inner product of p_c and e_n in L_2. Same holds for coefficients $b_n^{p_s}$.

Extension of this approach to more than one dimension is possible, but quickly becomes computationally intractable. The only exception is when the joint pdf p_c and p_s is factorisable $p_c(x_1,\ldots,x_n) = p_c(x_1)\cdot\ldots\cdot p_c(x_d)$ and $p_s(y_1,\ldots,y_n) = p_s(x_1)\cdot\ldots\cdot p_s(x_d)$, in which case it can be easily shown that

$$\mathrm{MMD}^2[\mathcal{F},p_c,p_s] = \left(\sum_{n=0}^{\infty}(b_n^{p_c})^2\right)^d - 2\left(\sum_{n=0}^{\infty}b_n^{p_c}b_n^{p_s}\right)^d + \left(\sum_{n=0}^{\infty}(b_n^{p_s})^2\right)^d,$$

where $b_{p_s,n}$, $b_{p_s,n}$ are as above. This approach is used in Section 8.3.1 to calculate exact values of MMD for artificially generated data sets.

8.3. Experiments

This section presents two experiments. The first experiment shows the convergence rates of $\mathrm{MMD}_u^2(\mathcal{F},\mathbf{X},\mathbf{Y})$ on two selected probability distributions. In the second experiment, MMD is used to benchmark security of several popular steganographic techniques. Since all presented experiments use Gaussian kernel (8.2.8) with free parameter γ, before experiments are described, the influence of kernel parameters and data pre-processing on MMD is discussed.

Theorem 8.2.8 guarantees that if a universal kernel is used (Gaussian kernel is universal), $\mathrm{MMD}[\mathcal{F},p_c,p_s]=0$ if and only if $p_c=p_s$. Even though from this point of view the choice of the kernel parameter γ seems to be irrelevant, γ does have a major influence on the accuracy of estimate (8.2.15) of MMD from finite number of samples. If γ is large, the Gaussian kernel is very narrow and thus $k(x,x^{'})\approx 0$ (the discrete approximation to the RKHS "overfits" the data). On the other hand, a very small γ leads to a wide kernel and $k(x,x^{'})\approx 1$ (the approximation is not "pliable" enough). With this in mind, it is important to select γ so that the Gaussian kernel changes its value rapidly with data. "Median" heuristic [**59**] (also

d	γ	2×500	2×1000	2×5000	2×10000	2×50000	∞
1	1.081421	0.22%	0.35%	−0.15%	0.01%	−0.02%	$4.81\cdot 10^{-1}$
5	0.112546	0.65%	−0.03%	0.27%	0.13%	−0.08%	$2.10\cdot 10^{-1}$
10	0.053799	0.27%	−0.36%	−0.00%	−0.22%	−0.01%	$1.22\cdot 10^{-1}$
100	0.005040	0.64%	−0.78%	0.03%	0.02%	0.02%	$1.44\cdot 10^{-2}$
200	0.002504	−0.28%	0.10%	−0.19%	−0.08%	0.02%	$7.28\cdot 10^{-3}$
300	0.001670	−0.59%	0.27%	−0.10%	−0.23%	−0.08%	$4.87\cdot 10^{-3}$

TABLE 8.3. Relative error of sample $\mathrm{MMD}_u^2(\mathcal{F},\mathbf{X},\mathbf{Y})$ with Gaussian kernel between two d-dimensional multivariate Gaussian distributions $N(-\frac{1}{\sqrt{d}},\mathbf{I})$ and $N(\frac{1}{\sqrt{d}},\mathbf{I})$ calculated from l data samples in d-dimensional space. The rightmost column denoted as ∞ shows the true value of MMD. The second leftmost column shows the width of Gaussian kernel γ used to calculate the estimates and true value.

used in one-class SVMs, see Section (6.1.1)) tries to achieve this by setting $\gamma = \frac{1}{\eta^2}$, where η is the median of L_2 distances between samples. This selection ensures that the test statistics is sensitive to data[6].

The pre-processing of the data is another important factor in application of MMD on real world data. Preliminary experiments showed that when Gaussian kernel is used, data should be normalized, i.e., the data should have zero mean and unit variance. Normalization ensures that feature with large variance does not dominate features with small variance.

The pre-processing together with kernel and its parameters defines the metrics of RKHS through dot-product (8.2.5). In benchmarking of steganographic methods, it is desired to directly compare values provided by MMD. Therefore, RKHS should be fixed for all stego methods. In this dissertation, the parameters of normalization and the kernel width γ are always determined from the cover samples only to ensure that the RKHS is derived under same conditions for all benchmarked stego-methods.

8.3.1. Experiments on artificial data sets.

8.3.1.1. *Multinomial Gaussian distribution.* This section investigates the relative error of MMD estimates exactly under the same conditions as in Sections 8.1.2.1 and 8.1.2.2, where the accuracy of the kNN estimator of KL divergence between various multinomial Gaussian distributions was examined. The only difference in the experiment setting is the number of samples for estimates on Gaussians with converging means. While kNN estimators used 2×10^5 samples, MMD estimators used only 2×5000 samples (20-times less). The intent is to emphasize the speed of MMD's convergence. All MMD estimators used Gaussian kernel with γ set according to the median heuristics.

From Table 8.3, showing the relative error of MMD calculated from Gaussians with KL divergence equal to 2, can be seen that the convergence rate of MMD is remarkably stable with respect to dimensionality of the data. The relative error of the estimate calculated from 2×5000 examples is within 1%. Results presented in Table 8.4 confirm the stability of the MMD on multinomial Gaussian distributions with converging means, which is important for verification of steganographic security.

[6]Let be $\eta = (x-y)$ and $\gamma = \frac{1}{\eta^2}$, then for the derivative of the Gaussian kernel with respect to x holds $\frac{\partial}{\partial x}\big|_{\eta=x-y} \exp\left(-\gamma(x-y)^2\right) = -\frac{2(x-y)}{e\eta}\Big|_{\eta=x-y} \approx -0.73$. High value of the derivative means that Gaussian kernel changes its value rapidly for point close to median.

d	γ	$\mu = \frac{1}{2\cdot\sqrt{d}}$	$\mu = \frac{1}{4\cdot\sqrt{d}}$	$\mu = \frac{1}{8\cdot\sqrt{d}}$	$\mu = \frac{1}{16\cdot\sqrt{d}}$	$\mu = \frac{1}{32\cdot\sqrt{d}}$	$\mu = 0$
1	1.113204	1.99%	0.05%	−2.17%	−2.58%	−8.83%	$7.43 \cdot 10^{-4}$
10	0.053313	0.26%	−0.68%	0.94%	2.22%	−0.56%	$1.31 \cdot 10^{-4}$
100	0.005026	−0.62%	0.87%	−1.96%	0.55%	−5.58%	$1.50 \cdot 10^{-5}$

TABLE 8.4. Relative error of MMD estimators on multivariate Gaussian distributions with dimensionality $d \in \{1, 10, 100\}$ and converging means. Estimates were calculated from 2×5000 samples. The second leftmost column shows the width of Gaussian kernel used to calculate estimated in the same row.

d	γ	2×500	2×1000	2×5000	2×10000	2×50000	true value
1	297.370640	−0.31%	−0.48%	0.00%	0.01%	0.31%	1.495132
5	120.008106	−1.14%	−1.63%	−0.65%	−0.48%	−0.10%	0.841277
10	79.470536	3.43%	0.84%	0.72%	−1.28%	−0.14%	0.543060
100	8.245417	9.12%	−2.32%	−0.15%	0.27%	−0.54%	0.055831
200	3.909558	0.30%	0.60%	−0.19%	−0.10%	0.28%	0.025575
300	2.388880	−3.14%	−3.76%	−0.96%	−0.95%	0.24%	0.016119

TABLE 8.5. MMD calculated between two multi-dimensional Student distributions $p_c \sim p_{\mathrm{St}}(\lambda = 0.01, \nu = 2, \mu = 0)$ and $p_s \sim p_S(\lambda = 0.01, \nu = 2, \mu = \frac{0.1}{\sqrt{d}})$, where $d \in \{1, 5, 10, 100, 200, 300\}$ is the dimension of the problem.

8.3.1.2. *Multinomial Student distribution.* According to [**33, 35**], output of many quantitative steganalyzers for LSB embedding follows Student t-distribution

$$p_{\mathrm{St}}(x|\lambda, \nu, \mu) = \frac{\Gamma(\frac{\nu+1}{2})}{\lambda\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})}\left(1 + \frac{(x-\mu)^2}{\lambda^2\nu}\right)^{-\frac{\nu+1}{2}}$$

with parameters degrees of freedom $\nu \approx 2$ and scale factor $\lambda = 0.01$. Moreover, their error follows a shift hypothesis, which says that the distribution of the detector response depends on the payload α only in the form of a shift, so that the (additive) estimation error is independent of the true value.

With above assumptions in mind, the convergence rates of MMD was verified on multidimensional Student t-distributions with pdfs $p_c \sim p_{\mathrm{St}}(\lambda = 0.01, \nu = 2, \mu = 0)$ and $p_s \sim p_S(\lambda = 0.01, \nu = 2, \mu = \frac{0.1}{\sqrt{d}})$ with dimensions $d \in \{1, 5, 10, 100, 200, 300\}$. Table 8.5 shows relative errors calculated from 100 repetitions. As in the case of multinomial Gaussians, MMD quickly approaches the target value. Estimates made from $l = 5000$ samples are within 1% of the true value of MMD.

8.3.2. Benchmarking steganographic methods. In this section, MMD is used to compare statistical detectability of 10 JPEG steganographic algorithms using the 274-dimensional Merged feature set (Section 4.4). The focus here is on low payloads to observe if any of the tested steganographic techniques becomes undetectable (indistinguishable using finite sample MMD).

The use of MMD to benchmark the steganographic methods might seem to be ungrounded, because MMD does not provide any ordering of dissimilarity of probability distributions. To justify the use of MMD recall that higher value of MMD means that the mean of stego distribution projected in RKHS is farther from the mean of cover distribution. Since the mean of cover distribution does not change, larger distance indicates that the stego distribution deviates more and the method would be more detectable. This presumption is experimentally verified by comparing the results from MMD benchmark to results obtained by widely used benchmark based on reporting the probability of error $P_{\mathrm{Err}} = 0.5(P_{\mathrm{FP}} + P_{\mathrm{MD}})$ of targeted steganalyzer (SVM in this case).

The image database for the experiment was created from 6006 images of a wide variety of scenes from 22 different digital cameras acquired in the raw uncompressed format. The images were embedded with pseudo-random payloads of $5\%, 9\%, 10\%, 15\%$, and 20% bpac (bits per non-zero AC coefficient). The payloads 9% and 10% were chosen intentionally to see the effect of matrix embedding with Hamming codes (the 9% payload can be embedded with a more efficient code). The tested stego algorithms include F5 [**72**], –F5 [**21**], F5 without shrinkage [**21**] (nsF5), JP Hide&Seek[7], Model Based Steganography without deblocking [**56**] (MBS1), MMx [**37**], Steghide [**27**], Perturbed Quantization while double compressing [**20**](PQ) and its two modifications (PQt and PQe) as described in [**21**]. The cover images were prepared for every method as if zero message was embedded. The quality factor for the first seven methods was set to 70 and thus the cover images were single-compressed JPEGs with quality 70. Because the three versions of PQ produce double-compressed images, the covers were created by double-compressing the raw images with the same quality factors of 85 and 70.

The empirical estimates $\mathrm{MMD}_u^2[\mathcal{F}, \mathbf{X}, \mathbf{Y}]$ were calculated from $l = 3000$ examples $\mathbf{X}$ from the cover class and 3000 examples $\mathbf{Y}$ from the stego class embedded with a specific message length. In each trial, samples were always chosen so that each original raw image appeared either in $\mathbf{X}$ or in $\mathbf{Y}$ but never in both. The calculation of estimates was repeated 100 times with a different split of the 6000 images and average of MMD values was taken. As was already discussed above, in order to ensure that all MMD values will be calculated in the same RKHS space, normalization parameters and width of Gaussian kernel γ (set according to "median" heuristic) were determined only on cover samples.

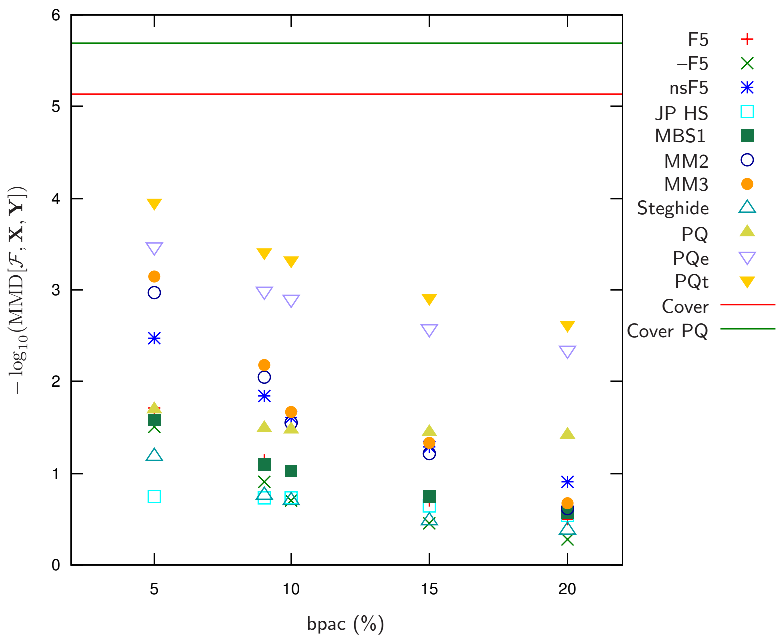
Figure 8.1 left shows $-\log_{10} \mathrm{MMD}[\mathcal{F}, \mathbf{X}, \mathbf{Y}]$ for 10 steganographic algorithms and 5 relative payloads. According to MMD, the PQ methods and the MMx are the least statistically detectable, while JP Hide&Seek, Steghide, and –F5 are the most detectable. F5 without shrinkage (nsF5) is the best algorithm that does not need side information (the raw image) at the embedder. The horizontal lines mark the value of MMD calculated from two disjoint samples of covers and thus indicate statistical undetectability with respect to the chosen feature set and database. Red line is for 70% quality JPEGs for the algorithms producing single-compressed images, while the green line is for double-compressed covers for PQ, PQe, and PQt. The error bars from the bootstrap are not shown, because the variances of MMD across different splits of the data set were too small to show in the graph.

Figure 8.1 right shows the $P_{\mathrm{Err}} = 0.5(P_{\mathrm{FP}} + P_{\mathrm{MD}})$ of soft-margin SVM targeted to given combination of steganographic algorithm and embedding rate. This benchmarking methodology was used in [**21, 63**]. Despite the fact that both benchmarking methods estimate steganographic security in a different way, the graphs appear to be consistent in the sense that stego methods with small MMD tend to have higher classification error and vice versa. This match between both steganographic benchmarks promotes the use of MMD, since it significantly reduces computational cost of the test, and removes some design decisions that need to be done in traditional testing that uses trained classifiers.

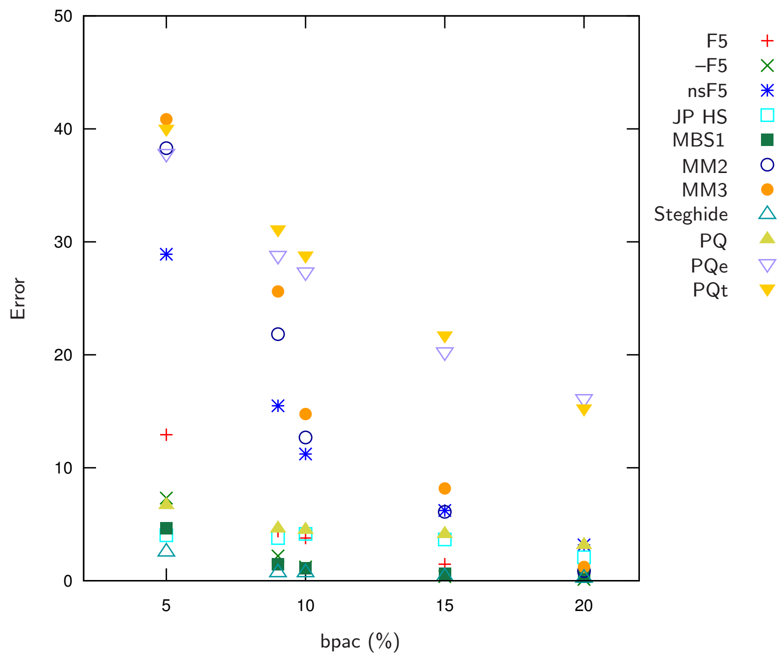
8.4. Conclusions

This part of the dissertation proposed a new method for benchmarking steganographic schemes. The project space $\mathcal{X}$ of feature set $\mathbf{f} : \mathcal{C} \mapsto \mathcal{X}$ is viewed as a simplified model of cover images. Consequently, security of steganographic schemes is measured with respect

[7]`linux01.gwdg.de/~alatham/stego.html`



(a) MMD



(b) SVM

FIGURE 8.1. MMD (left) and probability of error for an SVM (right) for 10 steganographic algorithms and 5 payloads. To obtain a better visual correspondence between the graphs, we show $-\log_{10}\mathrm{MMD}[\mathcal{F}, \mathbf{X}, \mathbf{Y}]$. The horizontal lines indicate the threshold of undetectability determined as MMD from two samples of covers. Algorithms with MMD close to the line are recognized as secure with respect to the given set of features.

to this simplified model. Statistical detectability of a given method for a fixed payload is evaluated by Maximum Mean Discrepancy (MMD) between the sample pdf of cover and stego features. MMD is proposed as a replacement of more theoretically grounded KL divergence, since the latter is difficult to estimate accurately from sparse data in high-dimensional spaces. The MMD offers a fast convergence rate with respect to the number of data samples even in high-dimensional spaces. Moreover, MMD's computational complexity is proportional to the square of the database size, which is relatively low. The MMD is used to replace classifiers involved in traditional benchmarks, and enables evaluating statistical detectability from the features themselves.

The benchmarking capability of MMD is demonstrated on 10 steganographic algorithms and compared to the benchmark based on SVM classifiers. The results of both benchmarks are surprisingly similar, which favors MMD because of its simplicity and low computational costs.

APPENDIX A

Support Vector Machines

This appendix reviews the basic principles of classification using Support Vector Machines (SVM), which are the tool of choice for most detectors described in this dissertation. This material is not meant as a detailed description of SVMs, but it's intention is to give the reader a good overview of SVMs and point out to difficulties of their applications. For a more detailed tutorial on SVMs the reader is referred to [**8, 69**].

The first, theoretic, part explains the main principles of Support Vector Machines on linearly separable problems, and extends the framework to problems that cannot be linearly separated. Theoretic part is finished by the description of kernelized Support Vector Machines. The second part of the appendix deals with practical issues, when SVMs are applied to real problems, and presents the most popular extensions for SVMs to classify into more than 2 classes.

PROBLEM A.0.1. [**Binary classification**] Let $\mathcal{X}$ be an arbitrary non-empty input space and $\mathcal{Y}$ be the label set $\mathcal{Y} = \{-1, +1\}$. Let assume *training set* $\{(x_i, y_i) | (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i \in \{1, \dots, l\}\}$ of l independent realizations of a random variable (x, y) with the joint pdf $P(x, y)$ on $\mathcal{X} \times \mathcal{Y}$ is available. The goal of binary classification is to find a function $f : \mathcal{X} \mapsto \mathcal{Y}$ that assigns a label to each $x \in \mathcal{X}$ and that makes as little errors as possible. f is evaluated using the risk functional

$$\mathcal{R}(f) = \int\limits_{\mathcal{X} \times \mathcal{Y}} u\left(-y f(x)\right) \mathrm{d}P(x, y),$$

where u is the step function

$$u(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0. \end{cases}$$

Clearly, $0 \leq \mathcal{R}(f) \leq 1$, and $\mathcal{R}(f) = 0$ when $f(x)$ correctly assigns the labels to all $x \in \mathcal{X}$ up to a set of measure zero (with respect to $P(x, y)$). At this point it is important to emphasize that unless the joint probability distribution $P(x, y)$ is known, it cannot be guaranteed that any *estimated* decision function is *optimal* (that it minimizes the risk functional $\mathcal{R}$).

A.1. Linear Support Vector Machines

A.1.1. Linearly separable training set. For the remainder of this appendix, it is assumed that the input space is $\mathcal{X} = \mathbb{R}^n$.

DEFINITION A.1.1. The training set is linearly separable, if there exists $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ such that decision function

$$f(\mathbf{x}) = \mathrm{sgn}((\mathbf{x} \cdot \mathbf{w}) + b), \tag{A.1.1}$$

has the *empirical* risk (error) on the training set $\mathcal{R}_{\mathrm{emp}}(f) = \frac{1}{l} \sum_{i=1}^{l} u\left(-y_i f(\mathbf{x}_i)\right) = 0$.

The function $f(\mathbf{x})$ classifies the point $\mathbf{x} \in \mathbb{R}^n$ based on which side of the hyperplane $\mathbf{x} \cdot \mathbf{w} + b$ the point $\mathbf{x}$ lies. Because the decision function is fully described by the *separating hyperplane*, terms decision function, hyperplane, or classifier are used interchangeably depending on the context.

If the training set is linearly separable, there exist infinitely many decision functions $f = \operatorname{sgn}((\mathbf{x} \cdot \mathbf{w}) + b)$ perfectly classifying the training set with $\mathcal{R}_{\text{emp}}(f) = 0$. To lower the probability of making incorrect decision on $\mathbf{x}$ not contained in the training set, SVM chooses the separating hyperplane with maximum distance from positive and negative training examples. This hyperplane, denoted by f^*, is uniquely defined. It can be found by solving the following optimization problem

$$[\mathbf{w}^*, b^*] = \arg \max_{\mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}} \{\min \{\|\mathbf{x} - \mathbf{x}_i\| | \mathbf{x} \in \mathbb{R}^n, (\mathbf{w} \cdot \mathbf{x}) - b = 0, i \in \{1, \ldots, l\}\}\} \tag{A.1.2}$$

subject to

$$y_i ((\mathbf{w} \cdot \mathbf{x}_i) - b) > 0, \forall i \in \{1, \ldots, l\}. \tag{A.1.3}$$

This optimization problem, however, is quite difficult to solve in practice. It can be reformulated as convex quadratic programming

$$[\mathbf{w}^*, b^*] = \arg \min_{\mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}} \frac{1}{2} \|\mathbf{w}\|^2 \tag{A.1.4}$$

subject to

$$y_i ((\mathbf{w} \cdot \mathbf{x}_i) - b) \geq 1, \forall i \in \{1, \ldots, l\}. \tag{A.1.5}$$

The latter problem (A.1.4) can be solved using standard quadratic programming tools (see the references in Section A.4).

The equivalence of the problem (A.1.2) and (A.1.4) might not be apparent on the first. Let assume that $[\mathbf{w}^*, b^*]$ is the solution of the first problem (A.1.2). Denoting $(\mathbf{x}^\circ, y^\circ), (\mathbf{x}^\bullet, y^\bullet)$ the closest example from positive, resp. negative class to the hyperplane, it has to hold

$$\begin{aligned} &\min \{\|\mathbf{x} - \mathbf{x}^\circ\| | \mathbf{x} \in \mathbb{R}^n, (\mathbf{w}^* \cdot \mathbf{x}) - b^* = 0, i \in \{1, \ldots, l\}\} = \\ &= \min \{\|\mathbf{x} - \mathbf{x}^\bullet\| | \mathbf{x} \in \mathbb{R}^n, (\mathbf{w}^* \cdot \mathbf{x}) - b^* = 0, i \in \{1, \ldots, l\}\}. \end{aligned}$$

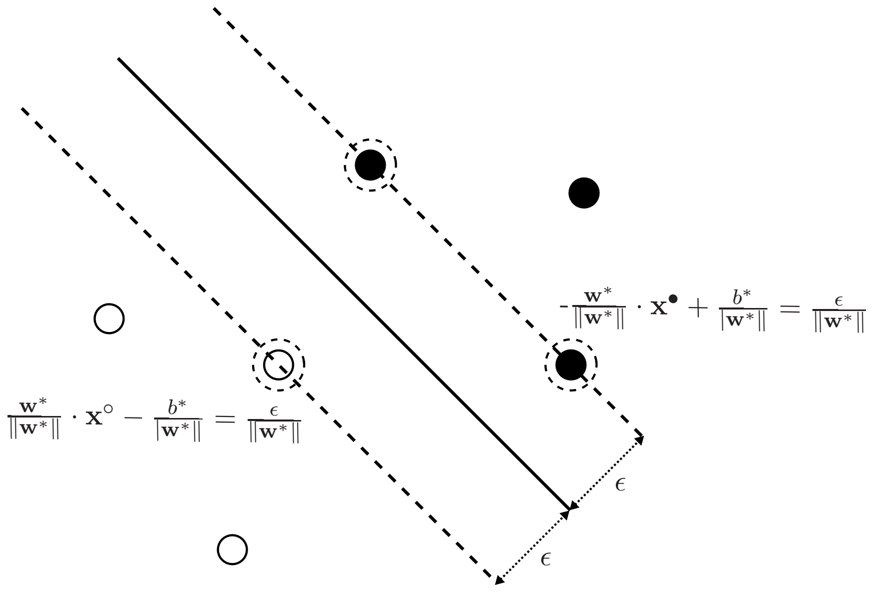
In other words the distances from the separating hyperplane to the closest point from each class have to be equal (see the Figure A.1). This is indeed true. If the distances were not equal, then we could move the hyperplane away from the closer class, which would increase the minimum (and margin) in (A.1.2) resulting in the contradiction with the optimality of the solution $[\mathbf{w}^*, b^*]$.

Even the closest examples $\mathbf{x}^\circ, \mathbf{x}^\bullet$ cannot lay on the separating hyperplane, due to the condition (A.1.3). Keeping in mind that $\mathbf{x}^\circ$ is from positive class, there has to exist $\epsilon > 0$ such that

$$\begin{aligned} \mathbf{w}^* \cdot \mathbf{x}^\circ - b^* &= +\epsilon, \\ \mathbf{w}^* \cdot \mathbf{x}^\bullet - b^* &= -\epsilon. \end{aligned} \tag{A.1.6}$$

By normalizing the last equations by ϵ, they can be rewritten as

$$\begin{aligned} \frac{\mathbf{w}^*}{\epsilon} \cdot \mathbf{x}^\circ - \frac{b^*}{\epsilon} &= +1, \\ \frac{\mathbf{w}^*}{\epsilon} \cdot \mathbf{x}^\bullet - \frac{b^*}{\epsilon} &= -1. \end{aligned} \tag{A.1.7}$$



$-\frac{\mathbf{w}^*}{\|\mathbf{w}^*\|} \cdot \mathbf{x}^\bullet + \frac{b^*}{\|\mathbf{w}^*\|} = \frac{\epsilon}{\|\mathbf{w}^*\|}$
$\frac{\mathbf{w}^*}{\|\mathbf{w}^*\|} \cdot \mathbf{x}^\circ - \frac{b^*}{\|\mathbf{w}^*\|} = \frac{\epsilon}{\|\mathbf{w}^*\|}$
ϵ
ϵ

FIGURE A.1. Example of linearly separable training set in $\mathcal{X} = \mathbb{R}^2$. Separating hyperplane (thick solid black line) is defined by the examples identified by dashed circle (called support vectors). Notice that other examples do not affect the solution of the optimization problem. By using notation borrowed from the text, for points on the separating hyperplane holds $\mathbf{w}^* \cdot \mathbf{x} - b^* = 0$, for points on dashed lines (support vectors) holds $\frac{\mathbf{w}^*}{\epsilon} \cdot \mathbf{x}^\circ - \frac{b^*}{\epsilon} = +1$ and $\frac{\mathbf{w}^*}{\epsilon} \cdot \mathbf{x}^\circ - \frac{b^*}{\epsilon} = +1$. For distance between support vectors and separating hyperplane holds $\left| \frac{\mathbf{w}^*}{\|\mathbf{w}^*\|} \cdot \mathbf{x} - \frac{b^*}{\|\mathbf{w}^*\|} \right| = \frac{\epsilon}{\|\mathbf{w}^*\|}$.

Because $\mathbf{x}^\circ, \mathbf{x}^\bullet$ are the closest examples, the remaining examples have to be farther away from the separating hyperplane. Consequently, the hyperplane $[\frac{\mathbf{w}^*}{\epsilon}, \frac{b^*}{\epsilon}]$ satisfies conditions (A.1.5). The margin between classes is equal to the sum of the distance of points $\mathbf{x}^\circ, \mathbf{x}^\bullet$ from the separating hyperplane $[\frac{\mathbf{w}^*}{\epsilon}, \frac{b^*}{\epsilon}]$. From (A.1.7) we get $\frac{\mathbf{w}^*}{\epsilon} \cdot (\mathbf{x}^\circ - \mathbf{x}^\bullet) = 2$, and after the normalization by $\left\| \frac{\mathbf{w}^*}{\epsilon} \right\|$ (distance of the point $\mathbf{x}$ from the hyperplane $[\mathbf{w}, b]$ is equal to $\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x} - \frac{b}{\|\mathbf{w}\|}$), we get the margin of $[\mathbf{w}^*, b^*]$ to be $\frac{\mathbf{w}^*}{\|\mathbf{w}^*\|} \cdot (\mathbf{x}^\circ - \mathbf{x}^\bullet) = \frac{2\epsilon}{\|\mathbf{w}^*\|}$. Naturally, maximizing margin $\frac{2\epsilon}{\|\mathbf{w}\|}$ is the same as minimizing $\frac{\|\mathbf{w}\|^2}{2}$ in (A.1.4), which finishes the proof of the equivalence of solutions of (A.1.2) and (A.1.4).

A.1.2. Non-separable training set. The linearly separable training set is a rather theoretic assumption. Real data are generally not linearly separable due to non-linearity of the practical problems and noise present in the measurement. Therefore, a mechanism that would admit classification errors on the training set is needed.

If the training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$ cannot be separated by a hyperplane without error, than the classifier f' should minimize the number of errors $\sum_{i=1}^{l} u\left(-y_i f'(\mathbf{x}_i)\right)$ on the training set. If all examples incorrectly classified by f' are excluded from the training set, $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$, then the training set becomes linearly separable and a maximum margin classifier f^* can be found in the same way, as in the previous section. The classifier f^* has the following important properties. First, its empirical risk on the training set is minimal. Second, it has maximum distance from correctly classified training examples.

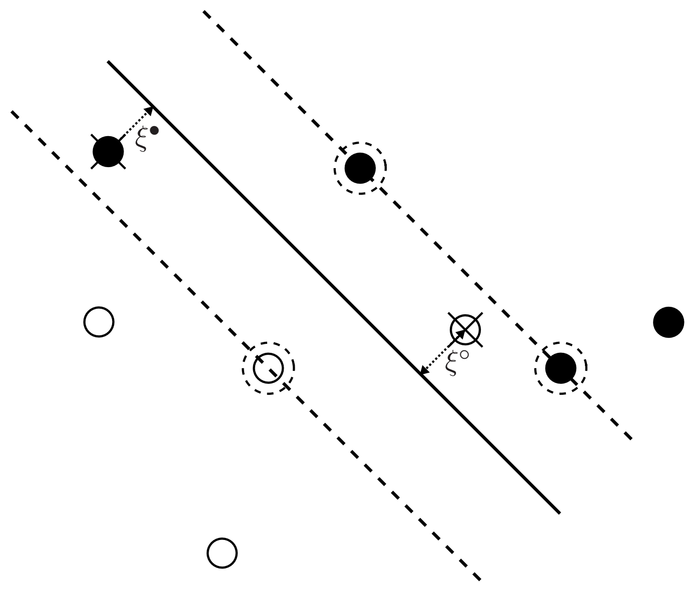


FIGURE A.2. Example of linearly non-separable training set on $\mathcal{X} = \mathbb{R}^2$. Separating hyperplane (thick solid black line) is defined by the examples identified by support vectors identified by dashed circle. Incorrectly classified examples are identified by cross $\times$ together with slack variables $\xi^\bullet, \xi^\circ > 0$.

The classifier f^* with maximum margin and minimal loss $\mathcal{R}_{\text{emp}}(f^*)$ can be found by solving the following optimization problem

$$[\mathbf{w}^*, b^*] = \arg\min_{w,b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C \cdot \sum_{i=1}^{l} u(\xi_i) \tag{A.1.8}$$

subject to constraints

$$\begin{aligned} y_i\left((\mathbf{w}\cdot\mathbf{x}_i) - b\right) &\geq 1 - \xi_i, \forall i \in \{1,\ldots,l\}, \\ \xi_i &\geq 0, \forall i \in \{1,\ldots,l\}, \end{aligned}$$

for some suitably chosen value of the penalization constant C. The "slack" variables ξ_i in (A.1.8) measure the distance of incorrectly classified examples $\mathbf{x}_i$ from the separating hyperplane. Of course, ξ_i is zero if $\mathbf{x}_i$ is classified correctly.

Unfortunately, the optimization problem (A.1.8) is NP-complete. The complexity can be significantly reduced by replacing the step function $u(x)$ with the hinge loss function $h(x) = \max\{0, 1 + x\}$. Because $h(x)$ is convex and $h(x) \geq u(x)$ for $x \geq 0$, it transforms the problem (A.1.8) to a convex quadratic programming problem

$$[\mathbf{w}^*, b^*] = \arg\min_{w,b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C \cdot \sum_{i=1}^{l} \xi_i \tag{A.1.9}$$

subject to constraints

$$\begin{aligned} y_i\left((\mathbf{w}\cdot\mathbf{x}_i) - b\right) &\geq 1 - \xi_i, \forall i \in \{1,\ldots,l\}, \\ \xi_i &\geq 0, \forall i \in \{1,\ldots,l\}. \end{aligned}$$

Notice that the optimization problem (A.1.9) minimizes the *overall distance* of incorrectly classified training examples from the hyperplane instead of their number. Support vector

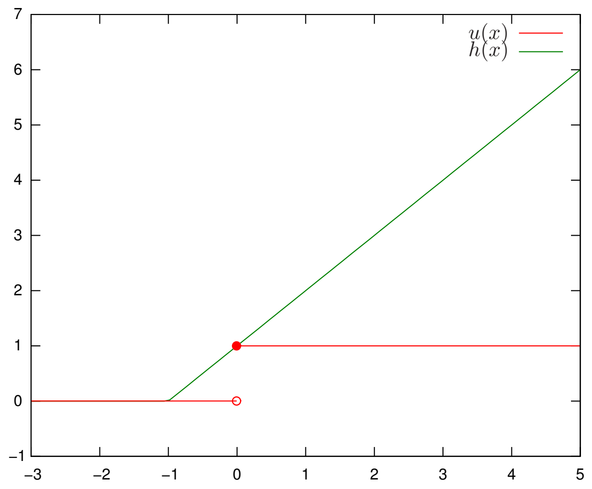


FIGURE A.3. Comparison of step function $u(x) = \begin{cases} 0 & x < 0 \\ 1 & \text{otherwise,} \end{cases}$ and its convex majority hinge loss $h(x) = \max\{0, 1 + x\}$.

machines that learn the decision function by solving (A.1.9) are called soft-margin Support Vector Machines (C-SVMs).

Even though the optimization problem (A.1.9) can be directly solved, in the course of training SVMs, it is usually solved in its dual form. The reason for this will become clear once we move to kernelized SVM in Section A.2. The constrained optimization problem (A.1.9) can be approached in a standard manner using Lagrange multipliers. The Lagrangian

$$L(\mathbf{w}, b, \xi, \mathbf{A}, \mathbf{r}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i - \sum_{i=1}^{l}\alpha_i\left[y_i((\mathbf{w}\cdot\mathbf{x}_i) - b) - 1 + \xi_i\right] - \sum_{i=1}^{l} r_i\xi_i \tag{A.1.10}$$

is formed with non-negative Lagrange multipliers $\mathbf{A} = (\alpha_1, \ldots, \alpha_l)$ and $\mathbf{r} = (r_1, \ldots, r_l)$. The solution of the problem (A.1.9) is in the saddle point of the Lagrangian $L(\mathbf{w}, b, \xi, \mathbf{A}, \mathbf{r})$—minimum with respect to variables $(\mathbf{w}, b, \xi)$ and maximum with respect to the Lagrange multipliers $(\mathbf{A}, \mathbf{r})$.

The Karush-Kuhn-Tucker conditions for the minimum of $L(\mathbf{w}, b, \xi, \mathbf{A}, \mathbf{r})$ at the extrema point (labeled again with superscript $*$) are

$$\left.\frac{\partial L}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}^*} = \mathbf{w}^* - \sum_{i=1}^{l}\alpha_i y_i \mathbf{x}_i = 0, \tag{A.1.11}$$

$$\left.\frac{\partial L}{\partial b}\right|_{b=b^*} = \sum_{i}^{l}\alpha_i y_i = 0, \tag{A.1.12}$$

$$\left.\frac{\partial L}{\partial \xi}\right|_{\xi_i=\xi_i^*} = C - \alpha_i - r_i = 0, \quad \forall i \in \{1, \ldots, l\}, \tag{A.1.13}$$

$$\alpha_i^*\left[y_i((\mathbf{w}^*\cdot\mathbf{x}_i) - b^*) - 1 + \xi_i^*\right] = 0, \quad \forall i \in \{1, \ldots, l\}, \tag{A.1.14}$$

$$r_i^*\xi_i^* = 0, \quad \forall i \in \{1, \ldots, l\}, \tag{A.1.15}$$

$$r_i^* \geq 0, \quad \forall i \in \{1, \ldots, l\}, \tag{A.1.16}$$

$$\alpha_i^* \geq 0, \quad \forall i \in \{1, \ldots, l\}. \tag{A.1.17}$$

After substituting the conditions (A.1.11), (A.1.12), and (A.1.13) into the Lagrangian (A.1.10), the dual formulation of the problem is obtained:

$$\max_{\mathbf{A},\mathbf{r}} L(\mathbf{A},\mathbf{r}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{A.1.18}$$

subject to constraints

$$\begin{aligned} \sum_{i=1}^{l} \alpha_i y_i &= 0, \\ C &\geq \alpha_i \geq 0, \forall i \in \{1,\dots,l\}. \end{aligned}$$

Note that the formulation of the dual problem does not contain the Lagrange multipliers r_i.

The main advantage of solving the dual problem (A.1.18) over the primal problem (A.1.9) is that the complexity (measured by the number of free variables) of the dual problem depends on the number of training examples, while the complexity of the primal problem depends in the dimension of the input space $\mathcal{X}$. After introducing kernelized SVMs in Section A.2, it will be seen that the dimension of the primal problem can be much larger (even infinite) than the number of training examples.

Denoting again the solutions of the dual problem (A.1.18) with superscript $*$, we need to recover the solution of the primal problem (A.1.9), which is the pair $(\mathbf{w}^*, b^*)$, from $\mathbf{A}^* = \{\alpha_1,\dots,\alpha_l\}$. From the KKT condition (A.1.11), we can easily obtain the hyperplane normal $\mathbf{w}^*$ as

$$\mathbf{w}^* = \sum_{i=1}^{l} \alpha_i^* y_i \mathbf{x}_i.$$

The computation of the threshold b^* is derived from KKT conditions (A.1.14) and (A.1.15). Let us assume that there is α_i^* such that $0 < \alpha_i^* < C$ (there is always at least one). Since $\alpha_i^* > 0$, the KKT condition (A.1.14) yields to

$$y_i((\mathbf{w}^* \cdot \mathbf{x}_i) - b^*) - 1 + \xi_i^* = 0. \tag{A.1.19}$$

From the KKT condition (A.1.13) and $\alpha_i^* < C$ follows that $r_i^* > 0$. The KKT condition (A.1.15) tells us that if $r_i^* > 0$, than $\xi_i^* = 0$. Substituting $\xi_i^* = 0$ to (A.1.19) yields the following equation for the threshold

$$b^* = (\mathbf{w}^* \cdot \mathbf{x}_i) - y_i. \tag{A.1.20}$$

Even though in the theory one α_i^* is enough to calculate the threshold b^*, for numerical stability it is better to take b^* as an average

$$b^* = \frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}} (\mathbf{w}^* \cdot \mathbf{x}_j) - y_j, \tag{A.1.21}$$

where $\mathcal{J} = \{i \in \{1,\dots,l\} | 0 < \alpha_i^* < C\}$.

Solving either the primal or dual optimization problem is commonly called training of SVMs. Technically, any optimization library that includes a routine for the quadratic programming problem can be used. Most general-purpose libraries, however, are usually able to solve only small scale problems. Therefore, it is highly recommended to use algorithms developed specifically for SVMs, such as LibSVM `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

A.2. Kernelized Support Vector Machines

Linear Support Vector Machines described in the previous section can only implement linear decision functions f, which is rarely sufficient for real-world problems. This section presents an extension to SVMs to implement non-linear decision function.

The main idea is to map the *input space* $\mathcal{X}$, which is the space where the observed data live, to a different space $\mathcal{H}$, using a non-linear mapping $\phi : \mathcal{X} \mapsto \mathcal{H}$ and then find the separating hyperplane in $\mathcal{H}$ (for linear SVMs described above, $\mathcal{X} = \mathcal{H}$). The non-linearity introduced through the mapping ϕ allows implementation of a non-linear decision boundary in the input space $\mathcal{X}$ as a linear decision boundary in $\mathcal{H}$.

While the input space $\mathcal{X}$ is usually given by the nature of the application (it is the space of features extracted from images), the space $\mathcal{H}$ can be freely chosen as long as it satisfies the following two conditions

(1) $\mathcal{H}$ has to be a Hilbert space (a complete space endowed with a dot-product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$).
(2) There has to exist a positive definite function[1] $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ called *kernel*, so that $\forall x, x^{'} \in \mathcal{X},\ k(x, x^{'}) = \left\langle \phi(x), \phi(x^{'}) \right\rangle_{\mathcal{H}}$.

These conditions ensure that the maximum margin hyperplane exists in $\mathcal{H}$ and that it can be found by solving the dual problem (A.1.18). More about the feature spaces $\mathcal{H}$ can be found in Section 8.2.1.

The kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ can be understood as a similarity measure on the input space $\mathcal{X}$. One of the most popular kernel functions is the Gaussian kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|),$$

where $\gamma > 0$ is a parameter controlling the width of the kernel. The Hilbert space $\mathcal{H}$ induced by the Gaussian kernel has infinite dimension. Other popular kernel is the polynomial kernel of degree d, defined as $k(\mathbf{x}, \mathbf{x}') = (r + \gamma \mathbf{x} \cdot \mathbf{x}')^{d}$, the linear kernel $k(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}$, and the sigmoid kernel $k(\mathbf{x}, \mathbf{x}') = \tanh(r + \gamma \mathbf{x} \cdot \mathbf{x})$ with parameters γ, r.

Non-linear SVMs are implemented in the same way as in Section A.1.2, except now all operations should be carried out in the Hilbert space $\mathcal{H}$. Because the dimensionality of the feature space $\mathcal{H}$ can be infinite, the dual optimization problem (A.1.18) has to be used, because the dimensionality of the problem is determined by the cardinality of the training set. Fortunately, because $\mathbf{x}_i$ always appears in the dot-product, the dot-product is simply replaced with its kernel based expression $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$. This substitution, called the "kernel trick", is possible in all algorithms, where the calculation with data appears exclusively in the form of dot-products.

After this substitution is performed, the dual optimization problem (A.1.18) becomes

$$\begin{aligned} \max_{\mathbf{A}, \mathbf{r}} L(\mathbf{A}, \mathbf{r}) &= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \left\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \right\rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j k\left(\mathbf{x}_i, \mathbf{x}_j\right), \end{aligned} \tag{A.2.1}$$

[1] $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is positive definite iff $(\forall n \geq 1)$ $(\forall x_1, \ldots, x_n \in \mathcal{X})$ $(c_1, \ldots, c_n \in \mathbb{R}) \left(\sum_{i,j=1}^{n} c_i c_j k(x_i, x_j) \geq 0 \right)$.

with constraints

$$\begin{aligned} \sum_{i=1}^{l} \alpha_i y_i &= 0 \\ C &\geq \alpha_i \geq 0, \forall i \in \{1, \dots, l\}. \end{aligned}$$

As the constraints do not contain any vector manipulation, they stay the same. In the equation $\mathbf{w}^* = \sum_{i=1}^{l} \alpha_i^* y_i \phi(\mathbf{x}_i)$ of the optimal hyperplane $\mathbf{w}^*$, all manipulations are being done in $\mathcal{H}$ and cannot be converted to the input space $\mathcal{X}$. Fortunately, we do not need to know $\mathbf{w}^*$ explicitly, because the decision function (A.1.1) can be rewritten as

$$f(\mathbf{x}) = \langle \mathbf{w}^*, \phi(\mathbf{x}) \rangle_{\mathcal{H}} - b^* = \sum_{j=1}^{l} \alpha_j^* y_j \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}) \rangle_{\mathcal{H}} - b^* = \sum_{j=1}^{l} \alpha_j^* y_j k\left(\mathbf{x}_j, \mathbf{x}\right) - b^*.$$

By the same mechanism, the equation (A.1.21) for the threshold b^* becomes

$$b^* = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} (\mathbf{w}^* \cdot \mathbf{x}_j) - y_j = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} \sum_{i=1}^{l} \alpha_i^* y_i k\left(\mathbf{x}_i, \mathbf{x}_j\right) - y_j,$$

with $\mathcal{J} = \{i \in \{1, \dots, l\} | 0 < \alpha_i^* < C\}$ as before.

Soft margin Support Vector Machines were proved to converge to an optimal classifier minimizing the risk functional $\mathcal{R}(f)$ as the number of training examples increases [**65**].

A.3. Weighted Support Vector Machines

In steganalysis, it is important to control the false positive rate of the steganalyzer. The desired false positive rate can be achieved either by shifting the threshold b^* in appropriate direction, or better by introducing different penalization coefficients for false positives and missed detections. The latter approach is more computationally expensive, as will be seen further, but for right choice of penalization coefficients gives superior performance [**5**].

Denoting $\mathcal{I}^-, \mathcal{I}^+$ the indices of negative and positive examples with $y_i = -1$ and $y_i = 1$, the primal problem of weighted Support Vector Machines accepts the form

$$\min_{w,b,\xi} \frac{1}{2} \|\mathbf{w}\|_{\mathcal{H}}^2 + C^- \cdot \sum_{i \in \mathcal{I}^-} \xi_i + C^+ \cdot \sum_{i \in \mathcal{I}^+} \xi_i \tag{A.3.1}$$

subject to constraints

$$\begin{aligned} y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle_{\mathcal{H}} - b \right) &\geq 1 - \xi_i, \forall i \in \{1, \dots, l\}, \\ \xi_i &\geq 0, \forall i \in \{1, \dots, l\}, \end{aligned}$$

where $\| \cdot \|_{\mathcal{H}}$ denotes the norm in the space $\mathcal{H}$. If we compare the original primal problem with equal costs of both detection errors (A.1.9) with (A.3.1), it is clear how differently the costs are expressed. By adjusting the penalization costs C^+ and C^-, more importance on one or the other error type can be put.

Following the same steps as in Section A.1.2, the dual form of (A.3.1) can be derived.

$$\max_{\mathbf{A},\mathbf{r}} L(\mathbf{A}, \mathbf{r}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j k\left(\mathbf{x}_i, \mathbf{x}_j\right), \tag{A.3.2}$$

subject to constraints

$$\begin{aligned}\sum_{i=1}^{l}\alpha_i y_i &= 0\\ C^+ &\geq \alpha_i \geq 0, \forall i \in \mathcal{I}^+,\\ C^- &\geq \alpha_i \geq 0, \forall i \in \mathcal{I}^-.\end{aligned}$$

The dual problem of weighted SVMs (A.3.2) is again a convex quadratic programming problem, which is almost identical to (A.2.1), with the only exception that now the Lagrange multipliers $\mathbf{A} = (\alpha_1, \dots, \alpha_l)$ are bounded by different constants depending on what training example they correspond to (e.g., if the training example is a cover image or a stego image).

The equation for the threshold b^* becomes

$$b^* = \frac{1}{|\mathcal{J}^-| + |\mathcal{J}^+|}\sum_{j\in\mathcal{J}^-\cup\mathcal{J}^+}\sum_{i=1}^{l}\alpha_i^* y_i k\left(\mathbf{x}_i, \mathbf{x}_j\right) - y_j,$$

where $\mathcal{J}^- = \{i \in \mathcal{I}^- | 0 < \alpha_i^* < C^-\}$ and $\mathcal{J}^+ = \{i \in \mathcal{I}^+ | 0 < \alpha_i^* < C^+\}$.

The decision function of weighted SVM $f(\mathbf{x}) = \sum_{j=1}^{l}\alpha_j^* y_j k\left(\mathbf{x}_j, \mathbf{x}\right) - b^*$ remains unchanged.

A.4. Practical use of Support Vector Machines

The kernel type and its parameters as well as the penalization parameter(s) have a great impact on the accuracy of the classifier. Unfortunately, there is no universal methodology how to best select them. This section offers some guidelines that often give good results. This methodology was used throughout this dissertation, whenever a SVM has to be trained.

A.4.1. Scaling. First, the input data needs to be pre-processed. Assuming $\mathcal{X} = \mathbb{R}^n$, which is the case for steganalysis, the input data is scaled so that all elements of vectors $\mathbf{x}_i$ from the training set are in the range $[-1, +1]$. This scaling is very important as it ensures that features with large numeric values do not dominate features with small values. It also increases numerical stability of the learning algorithm.

A.4.2. Kernel selection. The next step is the selection of a proper kernel. Unless some side information about the solved problem is known, the Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|)$ is typically a good first choice. This kernel is flexible enough to solve many problems, yet it only has one free parameter in comparison to polynomial or sigmoid kernels, which depend on more free parameters. Moreover, for all values of $\gamma > 0$, the Gaussian kernel is positively definite.

A.4.3. Determining parameters. Before training, the kernel parameters (the width γ if we use the Gaussian kernel) and the penalization parameter(s) (C^+, C^-) need to be determined. A common way to do so is to carry out an exhaustive search on predefined points from a grid $\mathcal{G}$. At each point $(C^+, C^-, \gamma) \in \mathcal{G}$, the SVM is trained and its performance on *unknown* data is estimated. The estimated probability of false positives and missed detections of SVM trained with parameters (C^+, C^-, γ) is denoted as $\hat{P}_{\mathrm{FA}}(C^+, C^-, \gamma)$, $\hat{P}_{\mathrm{MD}}(C^+, C^-, \gamma)$, respectively. The parameters (C^+, C^-, γ) are then selected as a point from the grid $\mathcal{G}$ using either Bayesian or Neyman-Pearson approach (see below).

The requirement of estimating performance on unknown data is very important. It is easy to find (C^+, C^-, γ) so that the error on the training set is zero, but this classifier will most probably have high error on the unknown data (this problem is called overtraining or overfitting).

A popular way for estimating $\hat{P}_{\text{FA}}(C^+,C^-,\gamma)$ and $\hat{P}_{\text{MD}}(C^+,C^-,\gamma)$ on unknown data is a *k-fold cross-validation*. The available training examples are divided into k subsets of approximately equal size. Then, the union of $k-1$ subsets is used as a training set to train the SVM, while the remaining subset is used to estimate the error on "unknown" examples. This is repeated k times, each time with different subsets. Estimates $\hat{P}_{\text{FA}}(C^+,C^-,\gamma)$ and $\hat{P}_{\text{MD}}(C^+,C^-,\gamma)$ are calculated as averages of error rates obtained on each fold. An extreme case of k-fold cross-validation is when number of folds k is equal to number of training examples $l-1$. Because the training is performed on $l-1$ examples and estimation is done on 1 example, this method is called *leave-one-out* cross-validation. Leave-one-out cross-validation is very computationally expensive and is used only in cases, when the number of examples in the training set is small. It has been showed that leave-one-out cross-validation provides an almost asymptotically unbiased estimate of the generalization error of SVM classifier (probability of error on sample not present in the training set) trained on $l-1$ examples.

Bayesian setting. When the costs of both errors are known and equal to w^+ and w^- for an error on positive and negative classes, weighted SVMs can be used with the Bayesian approach by minimizing the total cost

$$w^-p^-P_{\text{FA}} + w^+p^+P_{\text{MD}},$$

where p^- and p^+ are a priory probabilities of the negative and positive classes. In this case, the search for the parameters can be described for example as

$$\begin{aligned}(C^+,C^-,\gamma) &= \arg\min_{(C^+,C^-,\gamma)\in\mathcal{G}} w^-p^-\hat{P}_{\text{FA}}(C^+,C^-,\gamma) + w^+p^+\hat{P}_{\text{MD}}(C^+,C^-,\gamma),\\ \mathcal{G} &= \left\{(w^-p^-2^i, w^+p^+2^i, 2^j)|i\in\{-3,\dots,9\}, j\in\{-5,\dots,3\}\right\}.\end{aligned}$$

Note that even though the grid $\mathcal{G}$ has 3 dimensions, its effective dimension is 2 because the ratio $C^+/C^- = w^+p^+/w^-p^-$ stays constant.

Neyman-Pearson setting. Neyman-Pearson imposes upper bound on the probability of false alarms $P_{\text{FA}} \le \alpha < 1$ and minimizes the probability of missed detection P_{MD}. The search for (C^+,C^-,γ) becomes

$$\begin{aligned}(C^+,C^-,\gamma) &= \arg\min_{\substack{(C^+,C^-,\gamma)\in\mathcal{G},\\ \hat{P}_{\text{FA}}(C^+,C^-,\gamma)\le\alpha}} \hat{P}_{\text{MD}}(C^+,C^-,\gamma),\\ \mathcal{G} &= \left\{(2^i,2^j,2^k)|i,j\in\{-3,\dots,9\}, k\in\{-5,\dots,3\}\right\}.\end{aligned}$$

In this case, the grid $\mathcal{G}$ has the effective dimension 3, which makes the search computationally expensive. Frequently, sub-optimal search is used to alleviate the computational complexity, such as [**11**].

On the course of developing blind steganalyzer, the search is frequently performed on "unbounded" grid

$$\mathcal{G} = \left\{(2^i,2^i,2^j)|i\in\mathbb{Z}, j\in\mathbb{Z}\right\} \tag{A.4.1}$$

in Bayesian setting with equal weights and prior probabilities ($w^-p^- = w^+p^+$). The search on the grid A.4.1 exploits the observation that on most practical problems, the error surface of SVMs estimated using cross-validation is convex. The grid-search starts by evaluating all points from some small sub-grid $\mathcal{G}' \subset \mathcal{G}$. After all points are evaluated, the algorithm checks if the best point determined by the smallest cross-validation error $P_{err} = P_{FP} + P_{MD}$ is at the boundary of the sub-grid $\mathcal{G}'$. If so, the sub-grid is enlarged in the direction perpendicular to the boundary the best point laid on. The algorithm keeps doing this until the best point ends up within the explored sub-grid (not on the boundary), or the cross-validation error

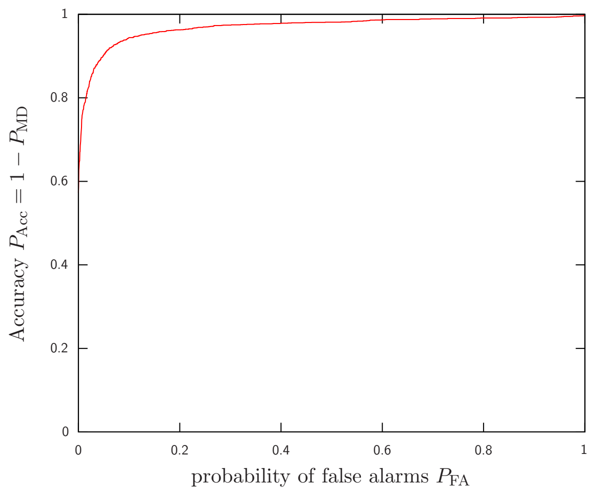


FIGURE A.1. Example of ROC curve created by varying threshold b^* of Support Vector Machine.

does not improve over a specified number of iterations. This simple algorithm ensures that the distance between the best point and the optimal point is small (within the size of the grid) under the convexity assumption.

A.4.4. Final training. After the kernel and its parameters including the penalization parameter(s) C are determined, the whole training set is used to train the final SVM. This process will determine the vector $(\alpha_1^*, \ldots, \alpha_l^*)$ and b^*. The decision function of the final SVM is $f(\mathbf{x}) = \sum_{j=1}^{l} \alpha_j^* y_j k(\mathbf{x}_j, \mathbf{x}) - b^*$.

A.4.5. Evaluating accuracy of Support Vector Machines.

A.4.5.1. *Receiver operating characteristic.* The Receiver Operating Characteristics (ROC) is a popular way to visualize performance of the set of classifiers $\mathcal{F}$. ROC shows the value of detection accuracy $P_{\text{Acc}} = 1 - P_{\text{MD}}$ as a function of the probability of false alarms (false positives) $\alpha = P_{\text{FP}}$. ROC is especially useful when one wants to compare different classes of classifiers, or when the costs of errors are not known in the time of training. It allows easy identification of regions with superior performance of given class $\mathcal{F}$.

Figure A.1 shows an example of ROC obtained by varying threshold b^* of trained SVM. The x-axis represents false positive rate P_{FP} and y-axis represents detection accuracy $P_{\text{Acc}} = 1 - P_{\text{MD}}$. Each point of the graph represents performance $(P_{\text{FP}}(f), P_{\text{Acc}}(f))$ of classifier $f \in \mathcal{F}$.

As in general $\mathcal{F}$ is a class of classifiers, there might exist classifiers $f, f' \in \mathcal{F}$ such that f has better detection accuracy than f' with the same false positive rate: $\left(P_{\text{Acc}}(f) > P_{\text{Acc}}(f')\right) \wedge \left(P_{\text{FP}}(f') = P_{\text{FP}}(f)\right)$. Since f clearly offers better performance than f', f' is not need to be displayed, because no one uses worse classifier when better is available. By displaying only the best classifier from $\mathcal{F}$ for a given false positive rate α, Receiver Operating Characteristic can be always displayed as a function of the false positive rate. This gives it the synonym *ROC curve*.

COROLLARY A.4.1. *For a given class of classifiers $f \in \mathcal{F}$, it is possible to construct classifier f' with accuracy $(P_{\text{FP}}(f'), P_{\text{Acc}}(f')) \in \mathcal{W}$, where $\mathcal{W}$ is the convex hull of the set* $\{(P_{\text{FP}}(f), P_{\text{Acc}}(f)) | f \in \mathcal{F}\}$.

Proof. The proof is straightforward, yet enlightening. For the convex hull $\mathcal{W}$ of the set $\{(P_{\mathrm{FP}}(f), P_{\mathrm{Acc}}(f)) | f \in \mathcal{F}\}$ holds

$$\mathcal{W} = \left\{ \sum_{i=1}^{n} \lambda_i (P_{\mathrm{FP}}(f_i), P_{\mathrm{Acc}}(f_i)) | n \in \mathbb{N}, \lambda_i \geq 0, \sum_{i=1}^{n} \lambda_i = 1, f_i \in \mathcal{F} \right\}. \tag{A.4.2}$$

Let suppose that one wants classifier with performance $(P^{'}_{\mathrm{FP}}, P^{'}_{\mathrm{Acc}}) \in \mathcal{W}$. Than there either exists classifier $f^{'} \in \mathcal{F}$ with performance $(P_{\mathrm{FP}}(f^{'}), P_{\mathrm{Acc}}(f^{'})) = (P^{'}_{\mathrm{FP}}, P^{'}_{\mathrm{Acc}})$, or there exist $n \geq 2$ classifiers $f_i \in \mathcal{F}, i \in \{1, \ldots, n\}$ and convex combination $(\lambda_1, \ldots, \lambda_n) \in [0,1]^n$, such that

$$(P^{'}_{\mathrm{FP}}, P^{'}_{\mathrm{Acc}}) = \sum_{i=1}^{n} \lambda_i (P_{\mathrm{FP}}(f_i), P_{\mathrm{Acc}}(f_i)).$$

Since the former case is trivial, the latter case is elaborated further by a constructing randomized classifier $f^{'}$. Every time the randomized classifier $f^{'}$ is making decision, it randomly selects classifier f_i with probability λ_i and returns the answer of f_i. It is obvious that for the accuracy of $f^{'}$ holds $\sum_{i=1}^{n} \lambda_i (P_{\mathrm{FP}}(f_i), P_{\mathrm{Acc}}(f_i))$, which is equal to the desired performance $(P^{'}_{\mathrm{FP}}, P^{'}_{\mathrm{Acc}})$. □

The important results of corollary A.4.1 is that it is always possible to augment the class of classifiers $\mathcal{F}$ by randomized classifiers, such that the ROC curve of the expanded class is *concave*. This result is frequently interpreted as that the ROC curve is always concave.

In the context of SVM, the calculation of ROC curve differs according to the definition of the classifier class $\mathcal{F}$. Most frequently, the class $\mathcal{F}$ comprises of a SVM trained with fixed parameters (kernel parameters and penalization constant(s) (C^{+}, C^{-})) with varying threshold b^{*}. It is clear that by changing threshold b^{*}, all values of P_{FP} (or detection accuracy P_{Acc}) can be reached. The advantage of this definition of class $\mathcal{F}$ is that the fast calculation of ROC curve, which can be done according to the algorithm 1.

The class $\mathcal{F}$ comprising of SVMs with varying parameters (penalization constant(s) (C^{+}, C^{-}), kernel parameters, kernel, etc.), has better ROC than the class $\mathcal{F}$, where only threshold b^{*} varies. Unfortunately, this definition of the class $\mathcal{F}$ is almost never used in practice, because it is computationally very expensive (SVM has to be trained for every value of penalization parameters, kernel, etc.). A promising solution for reducing the computational cost in the case when value of penalization parameters C^{+} and C^{-} change and kernel is fixed was recently proposed [**5**]. Since this approach is well beyond the scope of this introduction, it is not presented here.

A.4.5.2. *Comparing Receiver operating characteristic.* While comparing detectors by means of comparing their ROC curves is preferable, it is rarely used in practice from several reasons. First, to draw whole ROC curve can be very costly, as was discussed above. Second, it can happen that the ROC curve of two classifiers intersects, which means that there exist false positive rates $\alpha^{'}$ and $\alpha^{''}$ such that $\left(P^{(1)}_{\mathrm{Acc}}(\alpha^{'}) > P^{(2)}_{\mathrm{Acc}}(\alpha^{'})\right) \wedge \left(P^{(1)}_{\mathrm{Acc}}(\alpha^{"}) < P^{(2)}_{\mathrm{Acc}}(\alpha^{"})\right)$. In this case, it is not clear, which classifier is better. Third, if many classifiers need to be compared, it is easier to compare single numbers instead of whole curves.

These issues can be partially resolved, if a functional assigning ROC curve a single number is used. In steganography and steganalysis, several measures were proposed.

- The area between the ROC curve and the diagonal line normalized so that $\rho = 0$ when the ROC coincides with the diagonal line and $\rho = 1$ for ROC curves

Algorithm 1 Pseudo-code for the quick calculation points ROC curve $\mathcal{S}$ when penalization constant(s) (C^+, C^-) and kernel parameters are fixed. $(\alpha_1^*, \dots, \alpha_l^*)$ and b^* denote parameters of the trained SVM.

```
Train SVM with fixed penalization and kernel parameters.
Calculate the distance of testing examples $(\hat{\mathbf{x}}_i, \hat{y}_i)$ from
the separating hyperplane as

    $$d_i = \sum_{j=1}^{l} \alpha_j^* \hat{y}_j k\left(\hat{\mathbf{x}}_j, \mathbf{x}_i\right) - b^*.$$

Sort the points according to the $d_i$ such that $d_1 > d_2 > \dots >$
$d_m$.
Set the initial probabilities to $P_{\text{FP}} = 0, P_{\text{Acc}} = 0$.
Set $\mathcal{S} =$
for(i=1;i<=m;i=i+1){
 if $(y_i > 0)${
 $P_{\text{Acc}} = P_{\text{Acc}} + \frac{1}{|\hat{\mathcal{J}}^+|}$
 }
 if $(y_i < 0)${
 $\mathcal{S} = \mathcal{S} \cup \{(P_{\text{FP}}, P_{\text{Acc}})\}$
 $P_{\text{FP}} = P_{\text{FP}} + \frac{1}{|\hat{\mathcal{J}}^-|}$
 }
}
```

corresponding to nearly perfect detectors. Mathematically,

$$\rho = 2\int_0^1 P_{\text{Acc}}(\alpha)\mathrm{d}\alpha - 1.$$

This quantity is sometimes called accuracy.

- The minimal total average detection error under equal priors $(\Pr(\mathbf{x} \sim p_c) = \Pr(\mathbf{x} \sim p_s))$

$$P_{\text{Err}} = \min_{\alpha \in [0,1]} \frac{1}{2}(\alpha + P_{\text{Acc}}(\alpha)).$$

The minimum is reached at a point where the tangent to the ROC has slope 1/2.

- The false alarm rate at probability of detection equal to $P_{\text{Acc}} = 1/2$

$$\alpha_{1/2} = P_{\text{Acc}}^{-1}(0.5).$$

- Detection accuracy P_{Acc} at false positive rate $\alpha = 0.01$ or $\alpha = 0.00$.

None of these quantities is completely satisfactory because of missing fundamental reasoning behind them. The value $P_{\text{Acc}}(0.01)$ is probably the most useful for steganalysis because, as explained above, what matters the most is the detector's false alarm probability.

A.5. Multi-classification

Support Vector Machines were originally proposed for a binary classification problems. How to effectively extend them to classify into more than two classes is not entirely clear and there is still an active research in this area. There are two types of approaches to the multi-classification. One breaks the multi-classification problem into several binary problems, while the other one solves the multi-classification problem directly. According to [**28**], where theoretical and experimental comparison of different approaches of both types was presented, approaches based on breaking the multi-classification problem to several binary problems are more suitable for practical applications. They are easier to implement and they can handle larger classification problems. Despite that all multi-classifiers in the dissertation are implemented by the "max-wins" approach, three most popular schemes are described in the rest of this section.

Up to the end of this Section, it is assumed that the training set for a k-class classification problem is in the form $\{(x_1, y_1), \ldots, (x_l, y_l)\}$, $x_i \in \mathcal{X}$, $y_i \in \{1, \ldots, k\}$.

A.5.1. One-against-all. One-against-all is one of the earliest extensions of SVMs to multi-classification problems. To classify into k- classes, it constructs k binary SVMs (w^i, b^i). The training set of i-th SVM uses examples from i-th class as positive examples and all the others are used as negative examples. Formally written, for the training set of i-th binary SVM $\left\{(x_1, y_1^i), \ldots, (x_l, y_l^i)\right\}$ holds

$$y_j^i = \begin{cases} +1 & y_j = i \\ -1 & y_j \neq i. \end{cases}$$

The individual binary SVM can be trained with distinct kernel and penalization parameters. Denoting ϕ^i the feature mapping function corresponding to kernel function $k^i(x', x'') = \left\langle \phi^i(x'), \phi^i(x'') \right\rangle_{\mathcal{H}_i}$ of the i-th SVM, the decision on the input vector $x \in \mathcal{X}$ of the one-against-all classifier is carried out according to

$$\arg \max_{i \in \{1,\ldots,k\}} w^i \cdot \phi^i(x) - b^i.$$

The training of k binary SVMs is carried out in the dual form, as was explained in Section A.2. Assuming that the training set of all binary SVMs contains all l examples, and that the complexity of training binary SVM is $O(l^3)$, then the complexity of the training of the one-against-all classifier is $O(k \cdot l^3)$.

A.5.2. One-against-one ("max-wins"). The extension to the multi-classification adopted in this dissertation is usually called one-against-all. Since the DAG-SVM method described below uses exactly the same set of binary classifiers with different decision algorithm, the synonym "max-wins" refers to the decision algorithm described here.

This method constructs $\binom{k}{2}$ binary classifiers for each pair of classes $r \neq s$, $r, s \in \{1, \ldots, k\}$. The training set of the binary classifier for classes r, s contains examples only from those two classes. Formally written, the binary classifier $(w^{r,s}, b^{r,s})$ is trained on the set

$$\{(x_i, +1) | y_i = r\} \cup \{(x_i, -1) | y_i = s\}.$$

The decision of this method is done by "max-wins" strategy. The input vector $x \in \mathcal{X}$ is presented to all $\binom{k}{2}$ classifiers and the number of wins of each class are counted. At the end, the class with the highest number of votes is selected as a winner. In the case of tie, the winner is either selected randomly from the classes with highest number of wins, or it is some user selected default class.

If the number of examples from each class is equal, than the complexity of training of the "max-wins" multi-classifier is equal to $O\left(\binom{k}{2} \cdot \left(\frac{2l}{k}\right)^3\right)$.

A.5.3. DAG-SVM. DAG-SVM [**52**] uses the same set of binary machines as the "max-wins" approach, but adopts different decision strategy. During decision, the sample x bubbles through a rooted binary directed acyclic graph (DAG) starting in its root node. The sample leaves the internal node either to the right or to the left, depending on the outcome of the classification performed in the node. When the sample reaches the

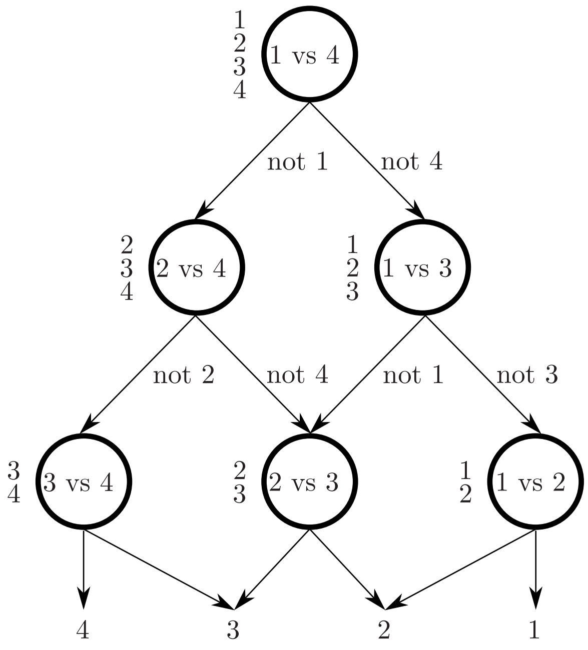


FIGURE A.1. The decision tree of DAG-SVM for finding the best out of four classes. Equivalent list of states is shown next to each node.

leaf, the value of the leaf is returned as the decision. Decision DAG for k classes contains $\binom{k}{2}$ internal nodes and k leaves, each indicating one class. An example of decision DAG is shown on Figure A.1.

The decision mechanism is equivalent to operating on the list, where each node eliminates one class from the list. In the beginning, the list is initialized with a list of all classes. In every step, the classification between the first and the last class in the list is performed, and the loosing class is removed from the list. When the list contains only one class, this last class is returned as the result.

The DAG-SVM offers several interesting advantages over the “max-wins” strategy. First, ambiguous cases when more than one class has the same maximum number of votes do not occur in DAG-SVM. Second, DAG-SVM makes the decision faster, as complexity of the decision making of DAG-SVM is $O(k)$, while the complexity of the decision making of the “max-wins” is $O(k^2)$. Third, theoretical bounds on the generalization error were established for DAG-SVM, which is not yet done for one-against-all and “max-wins”.

Despite these advantages, the experiments comparing DAG-SVM, “max-wins”, and one-against-all indicated that the “max-wins” is the most suitable for steganalysis.

Bibliography

[1] M. A. Arcones and E. Giné. On the bootstrap of u and v statistics. *The Annals of Statistics*, 20:655–674, 1992.

[2] I. Avcibas, M. Kharrazi, N. D. Memon, and B. Sankur. Image steganalysis with binary similarity measures. *EURASIP Journal on Applied Signal Processing*, 17:2749–2757, 2005.

[3] I. Avcibas, N. D. Memon, and B. Sankur. Steganalysis using image quality metrics. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents III*, volume 4314, pages 523–531, San Jose, CA, January 22–25, 2001.

[4] I. Avcibas, N. D. Memon, and B. Sankur. Image steganalysis with binary similarity measures. In *Proceedings IEEE, International Conference on Image Processing, ICIP 2002*, volume 3, pages 645–648, Rochester, NY, September 22–25, 2002.

[5] Francis Bach, David Heckerman, and Eric Horvitz. On the path to an ideal ROC curve: Considering cost asymmetry in learning classifiers. pages 9–16. Society for Artificial Intelligence and Statistics, 2005. (Available electronically at http://www.gatsby.ucl.ac.uk/aistats/).

[6] J. Beirlant, E. Dudewicz, L. Gyorfi, and E. van der Meulen. Non-parametric entropy estimation: An overview. *International Journal of Math. and Stat. Sci.*, 6:17–39, 1997.

[7] S. Boltz, E. Debreuve, and M. Barlaud. High-dimensional statistical distance for region-of-interest tracking: Application to combining a soft geometric constraint with radiometry. In *Proc. of CVPR*. IEEE Computer Society, 2007.

[8] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[9] C. Cachin. An information-theoretic model for steganography. In D. Aucsmith, editor, *Information Hiding, 2nd International Workshop*, volume 1525 of *Lecture Notes in Computer Science*, pages 306–318, Portland, OR, April 14–17, 1998. Springer-Verlag, New York.

[10] R. Chandramouli, M. Kharrazi, and N. D. Memon. Image steganography and steganalysis: Concepts and practice. In T. Kalker, I. J. Cox, and Y. Man Ro, editors, *Digital Watermarking, 2nd International Workshop*, volume 2939 of *Lecture Notes in Computer Science*, pages 35–49, Seoul, Korea, October 20–22, 2003. Springer-Verlag, New York.

[11] H. G. Chew, R. E. Bogner, and C.C. Lim. Dual-ν support vector machine with error rate and training size biasing. In *Proceeedings of Internation Conference on Acoustics, Speech, Signal Processing*, volume 2, pages 1269–1272, Salt Lake City, Utah, USA, 2001.

[12] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, Inc., 1991.

[13] R. M. Dudley. *Real analysis and probability.* Cambridge University Press, Cambridge, UK, 2002.

[14] J. Eggers, R. Bäuml, and B. Girod. A communications approach to steganography. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE Photonic West, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IV*, volume 4675, pages 26–37, San Jose, CA, January 21–24, 2002.

[15] Z. Fan and R. L. de Queiroz. Identification of bitmap compression history: JPEG detection and quantizer estimation. *IEEE Transactions on Image Processing*, 12(2):230–235, February 2003.

[16] H. Farid and L. Siwei. Detecting hidden messages using higher-order statistics and support vector machines. In F. A. P. Petitcolas, editor, *Information Hiding, 5th International Workshop*, volume 2578 of *Lecture Notes in Computer Science*, pages 340–354, Noordwijkerhout, The Netherlands, October 7–9, 2002. Springer-Verlag, New York.

[17] J. Fridrich. Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes. In J. Fridrich, editor, *Information Hiding, 6th International Workshop*, volume 3200 of *Lecture Notes in Computer Science*, pages 67–81, Toronto, Canada, May 23–25, 2004. Springer-Verlag, New York.

[18] J. Fridrich, P. Lisoněk, and D. Soukal. On steganographic embedding efficiency. In J. L. Camenisch, C. S. Collberg, N. F. Johnson, and P. Sallee, editors, *Information Hiding, 8th International Workshop*,

volume 4437 of *Lecture Notes in Computer Science*, pages 282–296, Alexandria, VA, July 10–12, 2006. Springer-Verlag, New York.

[19] J. Fridrich, M. Goljan, and D. Hogea. Steganalysis of JPEG images: Breaking the F5 algorithm. In *Information Hiding, 5th International Workshop*, volume 2578 of *Lecture Notes in Computer Science*, pages 310–323, Noordwijkerhout, The Netherlands, October 7–9, 2002. Springer-Verlag, New York.

[20] J. Fridrich, M. Goljan, and D. Soukal. Perturbed quantization steganography. *ACM Multimedia System Journal*, 11(2):98–107, 2005.

[21] J. Fridrich, T. Pevný, and J. Kodovský. Statistically undetectable JPEG steganography: Dead ends, challenges, and opportunities. In J. Dittmann and J. Fridrich, editors, *Proceedings of the 9th ACM Multimedia & Security Workshop*, pages 3–14, Dallas, TX, September 20–21, 2007.

[22] D. Fu, Y. Q. Shi, and Q. Su. A generalized Benford's law for JPEG coefficients and its applications in image forensics. In E. Delp and P. W. Wong, editors, *Proceeedings of SPIE Electronic Imaging, Security and Watermarking of Multimedia Contents IX*, volume 6505, pages 1L1–1L11, San Jose, CA, January 29 – February 1, 2007.

[23] M. Goljan, J. Fridrich, and T. Holotyak. New blind steganalysis and its implications. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, pages 1–13, San Jose, CA, January 16–19, 2006.

[24] Gretton, K. Borgwardt A., M. Rasch, B. Schoelkopf, and A. Smola. A kernel method for the two-sample-problem. Technical report, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, 2007. MPI Technical Report 157.

[25] Arthur Gretton, Karsten M. Borgwardt, Malte Rasch, Bernhard Schólkopf, and Alexander J. Smola. A kernel method for the two-sample-problem. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 513–520. MIT Press, Cambridge, MA, 2007.

[26] J. J. Harmsen and W. A. Pearlman. Steganalysis of additive noise modelable information hiding. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents V*, volume 5020, pages 131–142, Santa Clara, CA, January 21–24, 2003.

[27] S. Hetzl and P. Mutzel. A graph–theoretic approach to steganography. In J. Dittmann, S. Katzenbeisser, and A. Uhl, editors, *Communications and Multimedia Security, 9th IFIP TC-6 TC-11 International Conference, CMS 2005*, volume 3677 of *Lecture Notes in Computer Science*, pages 119–128, Salzburg, Austria, September 19–21, 2005.

[28] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 2001. http://citeseer.ist.psu.edu/hsu01comparison.html.

[29] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley-Interscience, 2001.

[30] A. L. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, 1989.

[31] A. Ker, T. Pevný, J. Kodovský, and J. Fridrich. The square root law of steganographic capacity. In A. Ker, J. Dittmann, and J. Fridrich, editors, *Proceedings of the 10th ACM Multimedia & Security Workshop*, Oxford, UK, September 22–23, 2008.

[32] A. D. Ker. Steganalysis of LSB matching in grayscale images. *IEEE Signal Processing Letters*, 12(6):441–444, June 2005.

[33] A. D. Ker. Batch steganography and pooled steganalysis. In J. L. Camenisch, C. S. Collberg, N. F. Johnson, and P. Sallee, editors, *Information Hiding, 8th International Workshop*, volume 4437 of *Lecture Notes in Computer Science*, Alexandria, VA, July 10–12, 2006. Springer-Verlag, New York.

[34] A. D. Ker. The ultimate steganalysis benchmark? In J. Dittmann and J. Fridrich, editors, *Proceedings of the 9th ACM Multimedia & Security Workshop*, pages 141–148, Dallas, TX, September 20–21, 2007.

[35] A. D. Ker and R. Böhme. A two-factor error model for quantitative steganalysis. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, pages 59–74, San Jose, CA, January 16–19, 2006.

[36] M. Kharrazi, H. T. Sencar, and N. D. Memon. Benchmarking steganographic and steganalytic techniques. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VII*, volume 5681, pages 252–263, San Jose, CA, January 16–20, 2005.

[37] Y. Kim, Z. Duric, and D. Richards. Modified matrix encoding technique for minimal distortion steganography. In J. L. Camenisch, C. S. Collberg, N. F. Johnson, and P. Sallee, editors, *Information Hiding, 8th International Workshop*, volume 4437 of *Lecture Notes in Computer Science*, pages 314–327, Alexandria, VA, July 10–12, 2006. Springer-Verlag, New York.

[38] J. Kodovský and J. Fridrich. On completeness of feature spaces in blind steganalysis. In A. Ker, J. Dittmann, and J. Fridrich, editors, *Proceedings of the 10th ACM Multimedia & Security Workshop*, Oxford, UK, September 22–23, 2008.

[39] M. Kutter and F. A. P. Petitcolas. A fair benchmark for image watermarking systems. *Security and Watermarking of Multimedia Contents*, Proc. SPIE - 3657:226–239, 1999.

[40] J. Lukáš and J. Fridrich. Estimation of primary quantization matrix in double compressed JPEG images. In *Proceedings of the Digital Forensic Research Workshop, DFRWS 2003*, Cleveland, OH, August 5–8, 2003.

[41] S. Lyu and H. Farid. Steganalysis using color wavelet statistics and one-class support vector machines. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306, pages 35–45, San Jose, CA, January 19–22, 2004.

[42] S. Lyu and H. Farid. Steganalysis using higher-order image statistics. *IEEE Transactions on Information Forensics and Security*, 1(1):111–119, 2006.

[43] Y. Miche, B. Roue, A. Lendasse, and P. Bas. A feature selection methodology for steganalysis. In B. Günsel, A. K. Jain, A. M. Tekalp, and B. Sankur, editors, *Multimedia Content Representation, Classification and Security, InternationalWorkshop*, volume 4105 of *Lecture Notes in Computer Science*, pages 49–56, Istanbul, Turkey, September 11–13, 2006. Springer-Verlag.

[44] P. Moulin, M. K. Mihcak, and G. I. Lin. An information–theoretic model for image watermarking and data hiding. In *Proceedings IEEE, International Conference on Image Processing, ICIP 2000*, volume 3, pages 667–670, Vancouver, Canada, September 10–13, 2000.

[45] A. Munoz and J. M. Moguerza. Estimation of high-density regions using one-class neighbor machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):476–480, 2006.

[46] H. Noda, M. Niimi, and E. Kawaguchi. Application of QIM with dead zone for histogram preserving JPEG steganography. In *Proceedings IEEE, International Conference on Image Processing, ICIP 2005*, pages II – 1082–5, Genova, Italy, September 11–14, 2005.

[47] W. Pennebaker and J. Mitchell. *JPEG: Still Image Data Compression Standard*. Van Norstrand Reinhold, 1993.

[48] F. Perez-Cruz. Kullback-leibler divergence estimation of continuous distributions. Workshop on Representations and Inference on Probability Distributions, NIPS 2007, December 8 2007.

[49] T. Pevný and J. Fridrich. Towards multi-class blind steganalyzer for JPEG images. In Mauro Barni, Ingemar J. Cox, Ton Kalker, and Hyoung Joong Kim, editors, *International Workshop on Digital Watermarking*, volume 3710 of *Lecture Notes in Computer Science*, Siena, Italy, September 15–17, 2005. Springer-Verlag, Berlin.

[50] T. Pevný and J. Fridrich. Determining the Stego Algorithm for JPEG Images. *Special Issue of IEE Proceedings — Information Security*, 153(3):75–139, 2006.

[51] T. Pevný and J. Fridrich. Merging Markov and DCT features for multi-class JPEG steganalysis. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, pages 3 1–3 14, San Jose, CA, January 29 – February 1, 2007.

[52] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report 98-14, Microsoft Research, Redmond, Washington.

[53] A. C. Popescu and H. Farid. Statistical tools for digital forensic. In J. Fridrich, editor, *Information Hiding, 6th International Workshop*, volume 3200 of *Lecture Notes in Computer Science*, pages 128–147, Toronto, Canada, May 23–25, 2004. Springer-Verlag, Berlin.

[54] A. C. Popescu and H. Farid. Exposing digital forgeries in color filter array interpolated images. *IEEE Transactions on Image Processing*, 53(10):3948–3959, 2005.

[55] N. Provos. Defending against statistical steganalysis. In *10th USENIX Security Symposium*, Proceedings of the ACM Symposium on Applied Computing, August 13–17, 2001.

[56] P. Sallee. Model-based steganography. In T. Kalker, I. J. Cox, and Y. Man Ro, editors, *Digital Watermarking, 2nd International Workshop*, volume 2939 of *Lecture Notes in Computer Science*, pages 154–167, Seoul, Korea, October 20–22, 2003. Springer-Verlag, New York.

[57] P. Sallee. Model-based methods for steganography and steganalysis. *International Journal of Image Graphics*, 5(1):167–190, 2005.

[58] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7), 2001.

[59] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press, 2001.

[60] R. J. Serfling. *Approximation Theorems of Mathematical Statistics*. Wiley-Interscience, 1980.

[61] Y. Q. Shi, C. Chen, and W. Chen. A Markov process based approach to effective attacking JPEG steganography. In J. L. Camenisch, C. S. Collberg, N. F. Johnson, and P. Sallee, editors, *Information Hiding, 8th International Workshop*, volume 4437 of *Lecture Notes in Computer Science*, pages 249–264, Alexandria, VA, July 10–12, 2006. Springer-Verlag, New York.

[62] H. Singh, N. Misra, V. Hnizdo, A. Fedorowicz, and E. Demchuk. Nearest neighbor estimates of entropy. *American Journal of Math. and Management Sciences*, 23:301–321, 2003.

[63] K. Solanki, A. Sarkar, and B. S. Manjunath. YASS: Yet another steganographic scheme that resists blind steganalysis. In T. Furon, F. Cayre, G. Doërr, and P. Bas, editors, *Information Hiding, 9th International Workshop*, Lecture Notes in Computer Science, pages 16–31, Saint Malo, France, June 11–13, 2007. Springer-Verlag, New York.

[64] K. Solanki, K. Sullivan, U. Madhow, B. S. Manjunath, and S. Chandrasekaran. Provably secure steganography: Achieving zero K-L divergence using statistical restoration. In *Proceedings IEEE, International Conference on Image Processing, ICIP 2006*, pages 125–128, Atlanta, GA, October 8–11, 2006.

[65] I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

[66] I. Steinwart, D. Hush, and C. Scovel. A classification framework for anomaly detection. *Journal of Machine Learning Research*, 6:211–232, 2005. Los Alamos National Laboratory Technical Report LA-UR-04-4716.

[67] I. Steinwart, D. Hush, and C. Scovel. Density level detection is classification. *Neural Information Processing Systems*, 17:1337–1344, 2005. Los Alamos National Laboratory Technical Report LA-UR-04-3768.

[68] I. Steinwart, D. Hush, and C. Scovel. An explicit description of the Reproducing Kernel Hilbert Spaces of Gaussian RBF kernels. *IEEE Transactions on Information Theory*, 52:4635–4643, 2006. Los Alamos National Laboratory Technical Report LA-UR-04-8274.

[69] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[70] Q. Wang, S. R. Kulkarni, and S. Verdu. A nearest-neighbor approach to estimating divergence between continuous random vectors. In *proceedings of IEEE International Symposium on Information Theory*, pages 242–246, 2002.

[71] Y. Wang and P. Moulin. Statistical modelling and steganalysis of DFT-based image steganography. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, pages 2 1–2 11, San Jose, CA, January 16–19, 2006.

[72] A. Westfeld. High capacity despite better steganalysis (F5 – a steganographic algorithm). In I. S. Moskowitz, editor, *Information Hiding, 4th International Workshop*, volume 2137 of *Lecture Notes in Computer Science*, pages 289–302, Pittsburgh, PA, April 25–27, 2001. Springer-Verlag, New York.

[73] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In *NIPS*, pages 668–674, 2000.

[74] G. Xuan, Y. Q. Shi, J. Gao, D. Zou, C. Yang, Z. Z. P. Chai, C. Chen, and W. Chen. Steganalysis based on multiple features formed by statistical moments of wavelet characteristic functions. In M. Barni, J. Herrera, S. Katzenbeisser, and F. Pérez-González, editors, *Information Hiding, 7th International Workshop*, volume 3727 of *Lecture Notes in Computer Science*, pages 262–277, Barcelona, Spain, June 6–8, 2005. Springer-Verlag, Berlin.